



TUGAS AKHIR - KI141502

SINKRONISASI BASIS DATA RELASIONAL MENGUNAKAN PARADIGMA PUBLISH-SUBSCRIBE

MUHAMMAD RIFATULLAH
NRP 5114100118

Dosen Pembimbing I
Rizky Januar Akbar, S.Kom., M.Eng.

Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom.,Phd.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018



TUGAS AKHIR - KI141502

SINKRONISASI BASIS DATA RELASIONAL MENGUNAKAN PARADIGMA PUBLISH-SUBSCRIBE

**MUHAMMAD RIFATULLAH
NRP 5114100118**

**Dosen Pembimbing I
Rizky Januar Akbar, S.Kom., M.Eng.**

**Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom.,Phd.**

**Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018**

(Halaman ini sengaja dikosongkan)



UNDERGRADUATE THESES - KI141502

RELATIONAL DATABASE SYSTEM SYNCHRONIZATION USING PUBLISH-SUBSCRIBE PARADIGM

**MUHAMMAD RIFATULLAH
NRP 5114100118**

First Advisor

Rizky Januar Akbar, S.Kom., M.Eng.

Second Advisor

Royyana Muslim Ijtihadie, S.Kom., M.Kom.,Phd.

Department of Informatics

Faculty of Information Technology and Communication

Institut Teknologi Sepuluh Nopember

Surabaya 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

SINKRONISASI BASIS DATA RELASIONAL MENGUNAKAN PARADIGMA PUBLISH-SUBSCRIBE

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada

Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

MUHAMMAD RIFATULLAH

NRP: 5114100118

Disetujui oleh Pembimbing Tugas Akhir

- 1 Rizky Januar Akbar, S.Kom.,
(198701032014041001) (Pembimbing 1)
- 2 Royyana Muslim Ijtihadie,
M.Kom.,Ph.D.
(197708242006041001) (Pembimbing 2)



SURABAYA

Januari, 2018

(Halaman ini sengaja dikosongkan)

SINKRONISASI BASIS DATA RELASIONAL MENGUNAKAN PARADIGMA PUBLISH-SUBSCRIBE

Nama Mahasiswa : MUHAMMAD RIFATULLAH
NRP : 5114100118
Jurusan : Teknik Informatika FTIF-ITS
Dosen Pembimbing 1 : Rizky Januar Akbar, S.Kom., M.Eng.
**Dosen Pembimbing 2 : Royyana Muslim Ijtihadie, S.Kom.,
M.Kom.,Phd.**

Abstrak

ITS mempunyai banyak sistem informasi. Namun, setiap sistem informasi mempunyai basis data masing-masing yang sebenarnya saling beririsan. Saat ini, setiap perubahan data harus dilakukan secara manual. Dengan jumlah data yang banyak dan perubahan yang masif, perubahan secara manual menjadi tidak efektif. Oleh sebab itu, dibutuhkan suatu sistem untuk sinkronisasi basis data. Pada tugas akhir ini, diajukan suatu metode untuk sinkronisasi basis data dengan pendekatan primary copy replication dan timestamp-based database synchronization dengan menggunakan paradigma publish-subscribe. Publish-subscribe digunakan untuk mengirimkan data dari tabel referensi ke tabel replika. Sinkronisasi diurutkan berdasarkan prioritas. Prioritas ditentukan oleh ketergantungan tabel tersebut ke tabel lain (dependensi). Tabel yang tidak mempunyai dependensi akan diproses terlebih dahulu.

Uji coba pada tugas akhir ini menggunakan data jabatan (stakeholder) dan unitnya. Jumlah data stakeholder adalah 100 data, sedangkan unit 184 data. Skenario uji coba terdiri dari beberapa kondisi seperti tidak mempunyai dependensi, dependensi terhadap diri sendiri (self-reference), dan dependensi terhadap tabel lain pada satu basis data.

Dari hasil uji coba, perubahan data pada tabel referensi bisa terdeteksi dan data berhasil dikirimkan ke tabel replika. Setelah data diproses oleh subscriber, data pada tabel referensi sama dengan tabel replika. Dengan menggunakan fitur RabbitMQ yaitu publisher confirms and message acknowledgements, data dapat dijamin sampai pada subscriber, walaupun broker atau subscriber mati.

Hasil uji coba menunjukkan bahwa metode yang diajukan mampu melakukan sinkronisasi basis data dengan benar walaupun tabel mempunyai dependensi atau terjadi gangguan pada koneksi agar konsistensi dan integritas data dapat terjaga.

Kata kunci: Basis Data, Sinkronisasi, Publish/Subscribe Pattern, Primary Copy Replication, Timestamp Based Database Synchronization.

RELATIONAL DATABASE SYSTEM SYNCHRONIZATION USING PUBLISH-SUBSCRIBE PARADIGM

Student's Name : MUHAMMAD RIFATULLAH
Student's ID : 5114100118
Department : Teknik Informatika FTIF-ITS
First Advisor : Rizky Januar Akbar, S.Kom., M.Eng.
Second Advisor : Royyana Muslim Ijtihadie, S.Kom.,
M.Kom.,Phd.

Abstract

ITS has many information systems. However, every system has its own database which often overlaps with other database. Until now, every data change must be committed manually. With the number of data and massive changes, manual commit is not effective. Therefore, a database synchronization system is needed. This research proposes a method for synchronizing database using primary copy replication and timestamp-based database synchronization with publish-subscribe paradigm. Publish-subscribe is used to transfer data from reference table to its replicas. Synchronization is ordered by priority. Priority is determined by table's dependency to another table. Table with no dependency will be processed first.

The testing in this research uses data of ITS stakeholders and units. There are 100 records for stakeholders data, and 184 records for unit data. The testing has different conditions for each scenario such as dependency free, self-referential relation and table-to-table relation.

The testing results show that data change can be detected and sent to replica successfully. After data is processed by subscriber, the data on reference table is equal to the data on replica table. With the help of the RabbitMQ capability of

publisher confirm and message acknowledgements, the data can be guaranteed to consume by the subscriber, even though the broker or subscriber dies.

The testing concluded that method proposed is capable of synchronizing table correctly even though it has dependency or network disruption while keeping consistency and data integrity.

Keywords: Database, Synchronization, Publish/Subscribe Pattern, Primary Copy Replication, Timestamp Based Database Synchronization.

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Alhamdulillahirabbil'alam, segala puji bagi Allah Swt yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

“SINKRONISASI BASIS DATA RELASIONAL MENGGUNAKAN PARADIGMA PUBLISH-SUBSCRIBE”

yang merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Selesaiannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah SWT atas izin dan rahmat Nya tugas akhir ini dapat diselesaikan.
2. Abah, Ibu, Mbak Zughro, Mbak Upik, Mas Ari, dan Dita beserta keluarga besar yang telah memberikan dukungan moral dan material, serta doa yang tak terhingga untuk penulis dalam mengerjakan tugas akhir ini.
3. Dita Adistia yang senantiasa sabar dan memberikan dukungan, doa dan motivasi, walaupun terkadang menyebalkan, serta menemani begadang demi pengerjaan tugas akhir ini.
4. Bapak Rizky Januar Akbar, S.Kom., M.Eng. dan Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom.,Phd. selaku pembimbing I dan pembimbing II yang telah membimbing dan membantu penulis serta memberikan motivasi dalam menyelesaikan Tugas Akhir ini.
5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya serta

staf karyawan Jurusan Teknik Informatika ITS yang telah memberikan bantuan demi kelancaran admisnistrasi penulis selama kuliah.

6. Ibu Anny Yuniarti, S.Kom., M.Comp.Sc selaku kepala Pusat Pengembangan DPTSI ITS yang telah membantu dan mendukung penulis.
7. Mas Ilham Gurat Adillion sebagai pembimbing bayangan 3 bidang penulisan, mas Hifnie Bilflash sebagai pembimbing bayangan 4 bidang teknis, serta Fatihah Ulya Hakiem sebagai pembimbing bayangan 5 bidang penjaminan mutu penulisan. Terimakasih atas dedikasi para pembimbing bayangan yang telah meluangkan waktu siang dan/atau malam di tengah kesibukan masing-masing, demi tercapainya tujuan mulia ini.
8. Kelompok pengungkap fakta: Elvacius, Vinsensia, Ivaldy, Aldo, Shafly alias Om, Dwika, Bimo, Aufar, Resha, Ikhsan, Faishal, Tosca, Neng Rara, dan Deni Ismail.
9. Shafly Naufal Adyanto alias om, teman kos terbaik yang selalu membantu walaupun pamrih.
10. Bapak, Ibu, Mas dan Mbak, teman-teman di DPTSI ITS yang sudah memotivasi dan mendukung penulis menyelesaikan tugas akhir ini.
11. Alumni OSIS MAN 1 Bandar Lampung, terutama angkatan 2012-2013.
12. Teman-teman TC 2014 yang senantiasa memotivasi penulis untuk menyelesaikan tugas akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapakan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Januari 2018

DAFTAR ISI

DAFTAR ISI.....	xiii
DAFTAR GAMBAR.....	xv
DAFTAR TABEL.....	xvii
DAFTAR KODE SUMBER	xix
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Permasalahan	3
1.4 Tujuan dan Manfaat	3
1.5 Metodologi	4
1.6 Sistematika Penulisan Laporan	5
BAB II TINJAUAN PUSTAKA	7
2.1 Basis Data Relasional.....	7
2.2 Sistem Informasi Manajemen.....	9
2.3 Replikasi dan Sinkronisasi	10
2.3.1 <i>Primary Copy Replication</i>	11
2.3.2 <i>Timestamp Based Database Synchronization</i>	11
2.4 Publish/Subscribe Pattern.....	13
2.4.1 <i>Message Queueing</i>	14
2.4.2 <i>Work Queue</i>	15
BAB III ANALISIS DAN PERANCANGAN SISTEM	17
3.1 Analisis.....	17
3.1.1 Analisis Permasalahan	17
3.1.2 Deskripsi Umum Perangkat Lunak.....	20
3.1.3 Arsitektur Sistem	22
3.1.4 Kebutuhan Fungsional Aplikasi.....	23
3.2 Perancangan	34
3.2.1 Perancangan Proses.....	34
3.2.2 Perancangan Basis Data.....	40
3.2.3 Perancangan Antarmuka	44
BAB IV IMPLEMENTASI.....	51
4.1 Lingkungan Implementasi.....	51
4.1.1 Lingkungan Implementasi Perangkat Keras	51

4.1.2	Lingkungan Implementasi Perangkat Lunak	51
4.2	Implementasi Basis Data	53
4.2.1	Implementasi Tabel conns	53
4.2.2	Implementasi Tabel queue_lists.....	53
4.2.3	Implementasi Tabel queue_log	54
4.2.4	Implementasi Tabel users	55
4.3	Implementasi Proses.....	55
4.3.1	Proses Menambah Koneksi.....	55
4.3.2	Proses Mengubah Koneksi.....	56
4.3.3	Proses Menghapus Koneksi	56
4.3.4	Proses Menambah Queue.....	57
4.3.5	Proses Mengubah Queue	58
4.3.6	Proses Menghapus Queue.....	58
4.3.7	Proses Sinkronisasi Basis Data	59
4.4	Implementasi Antarmuka	63
4.4.1	Antarmuka Halaman Menu Koneksi	63
4.4.2	Antarmuka Halaman Tambah Koneksi.....	64
4.4.3	Antarmuka Halaman Menu Daftar Queue	65
4.4.4	Antarmuka Halaman Tambah Queue.....	66
4.4.5	Antarmuka Halaman Mengubah Queue.....	68
4.4.6	Antarmuka Halaman Melihat Detail Queue	68
BAB V	HASIL UJI COBA DAN EVALUASI	70
5.1	Lingkungan Uji Coba.....	70
5.2	Dasar Pengujian.....	71
5.3	Pengujian Fungsionalitas.....	71
5.3.1	Pengujian Pengelolaan Koneksi Basis Data	71
5.3.2	Pengujian Pengelolaan Pengaturan Queue.....	72
5.3.3	Pengujian Pembaruan Prioritas Queue.....	75
5.3.4	Pengujian Sinkronisasi Basis Data.....	76
5.4	Evaluasi Pengujian	89
BAB VI	KESIMPULAN DAN SARAN.....	92
6.1	Kesimpulan.....	92
6.2	Saran.....	93
DAFTAR PUSTAKA		94
BIODATA PENULIS.....		96

DAFTAR GAMBAR

Gambar 2.1 Konsep sistem informasi	10
Gambar 2.2 Ilustrasi <i>Primary Copy Replication</i>	11
Gambar 2.3 <i>Publish/Subscribe Model</i>	15
Gambar 2.4 Proses Pengiriman Pesan dan Acknowledgments (ack)	16
Gambar 3.1 Arsitektur Sistem	22
Gambar 3.2 Diagram Kasus Penggunaan Sync ITS	23
Gambar 3.3 Diagram Kasus Penggunaan UC-001	25
Gambar 3.4 Diagram Kasus Penggunaan UC-002	30
Gambar 3.5 Diagram Kasus Penggunaan UC-003	31
Gambar 3.6 Diagram Kasus Penggunaan UC-004	33
Gambar 3.7 Contoh Prioritas Tabel.....	36
Gambar 3.8 Diagram Alur Sync ITS Publish Data	38
Gambar 3.9 Diagram Alur <i>Listener</i> Memproses Data.....	39
Gambar 3.10 PDM Sync ITS	40
Gambar 3.11 Rancangan Antarmuka Halaman Menu Koneksi ..	45
Gambar 3.12 Antarmuka Modal untuk Membuat koneksi baru..	46
Gambar 3.13 Rancangan Antarmuka Halaman Daftar <i>Queue</i>	47
Gambar 3.14 Rancangan Antarmuka untuk Menambah <i>Queue</i> ..	48
Gambar 3.15 Rancangan Antarmuka Halaman Melihat Detail Queue	50
Gambar 4.1 Pengaturan Basis Data Pada File .env	52
Gambar 4.2 Antarmuka Halaman Menu Koneksi	63
Gambar 4.3 Antarmuka untuk Menambah Koneksi.....	64
Gambar 4.4 Tampilan Info jika Nama Koneksi Sama	65
Gambar 4.5 Antarmuka Halaman Daftar Queue	65
Gambar 4.6 Antarmuka Halaman Tambah <i>Queue</i>	67
Gambar 4.7 Antarmuka Halaman Melihat Queue	69
Gambar 5.1 (a, b) Ilustrasi Pembaruan Prioritas <i>Queue</i>	76
Gambar 5.2 Pengaturan <i>Queue</i> untuk Pengujian.....	77
Gambar 5.3 Isi Tabel Unit (Referensi)	78
Gambar 5.4 Isi Tabel Unit (Replika) Sebelum Sinkronisasi	78
Gambar 5.5 Isi Tabel Unit (Replika) setelah Sinkronisasi	78

Gambar 5.6 Data pada Tabel Referensi.....	80
Gambar 5.7 Perubahan Data.....	80
Gambar 5.8 Hasil pada Tabel Replika.....	80
Gambar 5.9 Relasi Stakeholder dan Unit	81
Gambar 5.10 Data Tabel Stakeholder (Replika) sebelum Sinkronisasi	82
Gambar 5.11 Data Tabel Stakeholder (Referensi)	82
Gambar 5.12 Data Tabel Stakeholder (Replika) Setelah Sinkronisasi	82
Gambar 5.13 Data Stakeholder (Referensi).....	84
Gambar 5.14 Data Stakeholder (Replika) Setelah Sinkronisasi ..	84
Gambar 5.15 <i>Mapping</i> Tabel Referensi ke Tabel Replika	88
Gambar 5.16 Tampilan <i>Error Log</i> sebelum Sinkronisasi	88
Gambar 5.17 Hasil Deteksi setelah Sinkronisasi.....	88

DAFTAR TABEL

Tabel 2.1 Tabel Mahasiswa.....	9
Tabel 2.2 Tabel Mata Kuliah.....	9
Tabel 2.3 Tabel Nilai.....	9
Tabel 2.4 Contoh Tabel sync.....	13
Tabel 2.5 Contoh Tabel pengguna	13
Tabel 2.6 Hasil <i>Pulling</i> pada Tabel Pengguna	13
Tabel 3.1 Data ke $t = 0$ untuk (a) Data Pegawai Simpeg. (b) Data Pengguna e-Perkantoran.....	19
Tabel 3.2 Data ke $t=1$ untuk (a) Data Pegawai Simpeg. (b) Data Pengguna e-Perkantoran.....	19
Tabel 3.3 Data ke $t = 2$ untuk (a) Data Pegawai Simpeg. (b) Data Pengguna e-Perkantoran.....	19
Tabel 3.4 Data ke $t = 3$ untuk (a) Data Pegawai Simpeg. (b) Data Pengguna e-Perkantoran.....	20
Tabel 3.5 Deskripsi Kasus Penggunaan Sync ITS	24
Tabel 3.6 Rincian Alur Kasus Penggunaan UC-001	26
Tabel 3.7 Rincian Alur Kasus Penggunaan UC-001 (lanjutan)...	27
Tabel 3.8 Rincian Alur Kasus Penggunaan UC-002	28
Tabel 3.9 Rincian Alur Kasus Penggunaan UC-002 (lanjutan)...	29
Tabel 3.10 Rincian Alur Kasus Penggunaan UC-003	31
Tabel 3.11 Rincian Alur Kasus Penggunaan UC-004	32
Tabel 3.12 Rincian Alur Kasus Penggunaan UC-004 (lanjutan).	33
Tabel 3.13 Contoh table pengguna	36
Tabel 3.14 Atribut tabel conns	41
Tabel 3.15 Atribut pada tabel queue_lists	42
Tabel 3.16 Atribut Pada Tabel queue_log	43
Tabel 3.17 Atribut Pada Tabel users	44
Tabel 5.1 Tabel Spesifikasi Lingkungan Pengujian <i>Source</i>	70
Tabel 5.2 Tabel Spesifikasi Lingkungan Pengujian Server Sync ITS.....	70
Tabel 5.3 Tabel Spesifikasi Lingkungan Pengujian Target.....	71
Tabel 5.4 Skenario Pengujian Membuat Koneksi Baru	72
Tabel 5.5 Skenario Pengujian Mengubah Konfigurasi Koneksi ..	73

Tabel 5.6 Skenario Pengujian Menghapus Koneksi.....	73
Tabel 5.7 Skenario Pengujian Penambahan <i>Queue</i>	74
Tabel 5.8 Skenario Pengujian Mengubah <i>Queue</i>	74
Tabel 5.9 Skenario Pengujian Menghapus <i>Queue</i>	75
Tabel 5.10 Skenario Uji Coba 1	77
Tabel 5.11 Jumlah Pemrosesan <i>Queue</i> Uji Coba 1	79
Tabel 5.12 Skenario Uji Coba 2	80
Tabel 5.13 Jumlah Pemrosesan per Queue Uji Coba 2	81
Tabel 5.14 Skenario Uji Coba 3	82
Tabel 5.15 Jumlah Pemrosesan Stakeholder per <i>Queue</i> Uji Coba 3	83
Tabel 5.16 Skenario Uji Coba 4	84
Tabel 5.17 Jumlah Pemrosesan <i>Queue</i> pada Uji Coba 4.....	85
Tabel 5.18 Skenario Uji Coba 5	86
Tabel 5.19 Hasil Pemrosesan per Queue.....	86
Tabel 5.20 Skenario Uji Coba 6	87

DAFTAR KODE SUMBER

Kode Sumber 4.1 Implementasi Tabel conns.....	53
Kode Sumber 4.2 Implementasi Tabel queue_lists	54
Kode Sumber 4.3 Implementasi Tabel queue_log	54
Kode Sumber 4.4 Implementasi Tabel users	55
Kode Sumber 4.5 Implementasi Proses Menambah Koneksi	56
Kode Sumber 4.6 Implementasi Proses Mengubah Koneksi	56
Kode Sumber 4.7 Implementasi Proses Menghapus Koneksi	57
Kode Sumber 4.8 Implementasi Proses Menambah Queue	57
Kode Sumber 4.9 Implementasi Proses Mengubah Queue	58
Kode Sumber 4.10 Implementasi Proses Menghapus Queue.....	58
Kode Sumber 4.11 Perintah Laravel untuk Penjadwalan	59
Kode Sumber 4.12 Crontab untuk Penjadwalan Sinkronisasi	59
Kode Sumber 4.13 Perintah Eksekusi Sinkronisasi	59
Kode Sumber 4.14 Implementasi Pekerjaan SyncJob	60
Kode Sumber 4.15 Sync ITS Melakukan <i>Pulling</i> ke Tabel Referensi.....	60
Kode Sumber 4.16 Pengecekan <i>Self-Reference</i>	60
Kode Sumber 4.17 Proses <i>Chunking</i> Data	61
Kode Sumber 4.18 Sync ITS Mempublish Data ke RabbitMQ ..	61
Kode Sumber 4.19 Proses Perbarui Kolom last_sync	61
Kode Sumber 4.20 Proses Tambah Data oleh Listener	62
Kode Sumber 4.21 Proses Perbarui Data oleh Listener	62
Kode Sumber 4.22 Proses Pengiriman <i>Acknowledgements</i>	62
Kode Sumber 4.23 Proses Pengiriman <i>Negative Acknowledgements</i>	62
Kode Sumber 4.24 Proses Publish Ulang <i>Queue</i>	63

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

1.1 Latar Belakang

Direktorat Pengembangan dan Teknologi Sistem Informasi (DPTSI) merupakan direktorat yang bergerak di bidang pembuatan dan pengembangan sistem informasi di Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Sebagai pusat teknologi informasi, DPTSI bertanggungjawab terhadap aplikasi-aplikasi yang terdapat di ITS. DPTSI mengelola banyak aplikasi baik aplikasi web maupun aplikasi perangkat bergerak. Namun setiap aplikasi mempunyai basis data masing-masing yang sebenarnya saling berkaitan. Sebagai contoh, DPTSI mempunyai Sistem Kepegawaian (SIMPEG) yang di dalamnya terdapat data pegawai ITS. Apabila DPTSI akan membuat sistem baru yang membutuhkan data pegawai ITS, contohnya E-perkantoran, maka data pegawai dari SIMPEG akan diekspor ke sistem E-perkantoran. Data pegawai pada SIMPEG merupakan tabel referensi, sedangkan data pegawai pada E-perkantoran merupakan tabel replika. Kekurangan dari proses tersebut adalah saat terjadi perubahan data pada data pegawai, harus dilakukan manipulasi data di setiap aplikasi secara manual. Hal ini menjadi masalah apabila terjadi perubahan data dengan skala yang masif. Integritas dan konsistensi data menjadi tidak terjamin sehingga pada suatu saat dapat terjadi kondisi dimana tidak ada basis data yang dapat dijadikan referensi.

Untuk menangani masalah tersebut, dapat dilakukan sinkronisasi data. Pendekatan sinkronisasi data yang sudah diterapkan pada beberapa aplikasi di DPTSI adalah *linked server* [1]. *Linked server* adalah suatu metode yang memungkinkan *query* dari banyak *server* sekaligus. Cara tersebut merupakan cara yang paling mudah karena fitur *linked server* merupakan fitur SQL Server yang dapat dikonfigurasi dengan menggunakan *SQL Server Management Studio* atau *Transact-SQL*. *Linked server* menjamin konsistensi data. Namun, terdapat kekurangan seperti *workload* menjadi lebih besar karena transaksi data terjadi pada basis data

referensi yang mungkin diakses oleh banyak aplikasi. Performa sistem menurun karena membutuhkan waktu yang lebih lama untuk meng-*query* data dari *server* lain. Menurut Munoz dkk [2], terdapat pendekatan lain untuk replikasi dan sinkronisasi data yaitu *primary copy replication*. Pada teknik tersebut tabel pada basis data referensi direplikasikan ke basis data yang membutuhkan dengan hak akses *read-only*. Semua manipulasi data diarahkan dan dieksekusi pada basis data referensi, kecuali *query select*. Aplikasi tidak melakukan *query select* ke basis data referensi, melainkan dilakukan pada basis data lokal. Metode ini meningkatkan performa sistem secara keseluruhan karena beban basis data referensi terbagi pada basis data lokal.

Pada tugas akhir ini akan dibuat suatu aplikasi yang berperan sebagai *synchronizer* dengan nama Sync ITS. Sync ITS menggunakan pendekatan *primary copy replication* dan *timestamp based synchronization* dengan paradigma *publish/subscribe* [3] sebagai pola pendistribusian pesan (*message broker*). Daftar tabel referensi dan tabel replikanya tersimpan di dalam aplikasi ini. Pada setiap tabel terdapat suatu kolom untuk mencatat kapan perubahan terjadi. Ketika sinkronisasi dimulai, maka Sync ITS akan menarik (*pull*) data yang waktu perubahannya lebih dari waktu sinkronisasi terakhir pada tabel referensi. Lalu aplikasi akan memublikasikan pesan tersebut ke tabel replika dengan bantuan RabbitMQ. Data akan diterima oleh *listener*, lalu *listener* akan melakukan proses *read*, *create* dan *update* pada tabel replika. Sync ITS berperan sebagai *publisher*, sedangkan *listener* berperan sebagai *subscriber*. Untuk memastikan integritas data, *publisher* harus mendapat *acknowledgements* dari *listener*. Jikalau tidak ada *acknowledgements* [4], maka pesan dianggap gagal lalu *queue* akan dibuat ulang untuk dikirimkan kembali ke tabel replika. Untuk menghindari *infinite loop*, maka setiap *queue* mempunyai maksimal pengulangan, sehingga ketika pengulangan lebih dari maksimal pengulangan maka *queue* akan ditolak dan tidak dipublish ulang. Tugas akhir ini diharapkan dapat menyelesaikan

masalah pada manajemen data di ITS sehingga konsistensi dan integritas data dapat terjaga.

1.2 Rumusan Masalah

Rumusan masalah yang terdapat pada tugas akhir ini di adalah sebagai berikut:

1. Bagaimana cara mengimplementasikan sinkronisasi satu arah antar tabel referensi dan replikanya dengan menggunakan metode *primary copy replication* dan *timestamp-based database synchronization*?
2. Bagaimana cara mendistribusikan perubahan data dari tabel referensi ke tabel replika dengan menggunakan paradigma *publish/subscribe*?
3. Bagaimana menangani data yang gagal tersinkronisasi karena jaringan yang terputus?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini memiliki batasan sebagai berikut:

1. Sinkronisasi satu arah, yaitu sinkronisasi dari tabel referensi ke tabel replika.
2. Basis data yang digunakan adalah SQL Server.
3. Semua aplikasi yang terlibat seperti web server dan basis data harus menggunakan zona waktu yang sama.
4. Menggunakan RabbitMQ sebagai *message broker*.

1.4 Tujuan dan Manfaat

1. Tujuan
Tujuan dari pembuatan tugas akhir ini adalah mengembangkan sistem replikasi dan sinkronisasi data antar basis data relasional untuk kebutuhan ITS.
2. Manfaat
Manfaat yang diperoleh dari pembuatan tugas akhir ini adalah sinkronisasi data antar basis data referensi dengan

basis data replika di ITS untuk memudahkan DPTSI ITS dalam proses manajemen data dan menjaga konsistensi data.

1.5 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

A. Penyusunan Proposal

Proposal tugas akhir ini berisi rencana tugas akhir yang akan dikerjakan sebagai syarat untuk menyelesaikan studi dan meraih gelar Strata-1 Teknik Informatika. Terdapat penjelasan mengenai latar belakang pengambilan tema, rumusan masalah, batasan masalah, tujuan, dan manfaat dari rencana tugas akhir ini. Selain itu juga dijelaskan mengenai metode apa saja yang digunakan serta penjelasannya. Agar lebih mudah dipahami, penjelasan disertai dengan diagram alir.

B. Studi Literatur

Pada tahap ini dilakukan untuk mencari informasi dan studi literatur apa saja yang dapat dijadikan referensi untuk membantu pengerjaan tugas akhir ini. Informasi didapatkan dari buku dan literatur yang berhubungan dengan metode yang digunakan. Informasi yang dicari adalah literatur mengenai pendekatan yang bisa diaplikasikan untuk replikasi dan sinkronisasi basis data sesuai kebutuhan DPTSI ITS dan *publish/subscribe pattern*.

C. Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal Tugas Akhir.

Untuk membangun perangkat lunak yang telah dirancang sebelumnya, maka dilakukan implementasi dengan menggunakan suatu perangkat lunak yaitu *PHPStorm* sebagai IDE, *Navicat* sebagai *Database Client*, dan basis data SQL Server 2016.

D. Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba pada data yang telah dikumpulkan. Tahapan ini dimaksud untuk mengevaluasi kesesuaian data dan program serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

E. Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi mengenai pembuatan tugas akhir yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.6 Sistematika Penulisan Laporan

Buku tugas akhir ini disusun dengan tujuan untuk memberikan gambaran tentang pengerjaan tugas akhir yang telah dilakukan. Buku tugas akhir ini terbagi menjadi enam bab, yaitu:

A. Bab I. Pendahuluan

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.

B. Bab II. Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

C. Bab III. Analisis dan Perancangan Sistem

Bab ini berisi membahas tentang desain dan rancangan dari perangkat lunak. Rancangan dan desain meliputi data, proses, dan arsitektur.

D. Bab IV. Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *code* yang digunakan untuk proses implementasi.

E. Bab V. Hasil Uji Coba dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

F. Bab VI. Kesimpulan dan Saran

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

G. Daftar Pustaka

Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.

BAB II

TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam tugas akhir. Dasar teori yang dibahas di dalam bab ini adalah penjelasan basis data relasional, sistem informasi manajemen, replikasi dan sinkronisasi, metode *primary copy replication*, *timestamp based database synchronization*, serta paradigma *publish/subscribe*. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Basis Data Relasional

Basis data adalah sekumpulan data maupun deskripsi tentang data yang secara logis saling berhubungan untuk digunakan bersama dalam rangka memenuhi kebutuhan informasi dari suatu organisasi [5].

Basis data adalah kumpulan data yang berhubungan secara logis dan deskripsi data tersebut, yang dirancang untuk memenuhi informasi yang dibutuhkan oleh suatu organisasi. Artinya, basis data merupakan tempat penyimpanan data besar yang dapat digunakan oleh banyak pengguna. Seluruh item basis data tidak lagi dimiliki oleh satu departemen, tetapi menjadi sumber daya perusahaan yang dapat digunakan bersama [6].

Berikut adalah komponen-komponen yang terdapat pada basis data:

1. Data merupakan komponen terpenting dari suatu basis data. Data merupakan fakta atau observasi mentah yang biasanya mengenai fenomena fisik atau transaksi bisnis [6]. Pengumpulan data dilakukan secara sistematis menurut struktur basis data tersebut.

2. Tabel adalah kumpulan dari suatu *field* dan *record*. Biasanya *field* ditujukan dalam bentuk kolom dan *record* dalam bentuk baris.
3. *Record* adalah sebuah data yang isinya merupakan suatu kesatuan, setiap keterangan yang mencakup kesatuan tersebut dinamakan satu *record*. Setiap *record* diberi nomor urut yang disebut nomor *record*.
4. *Field* atau kolom atau atribut merupakan kesatuan terkecil dari informasi dalam sebuah basis data. Kumpulan *field* yang saling berkaitan akan membentuk *record*.
5. *Primary Key* adalah suatu kolom (*field*) yang menjadi titik acuan pada sebuah tabel, bersifat unik dalam artian tidak ada satu nilai pun yang sama atau kembar.
6. *Foreign Key* adalah suatu kolom (*field*) yang digunakan sebagai acuan untuk berelasi dengan tabel lain.
7. *Index* adalah struktur data secara fisik yang digunakan untuk optimalisasi pemrosesan data dan mempercepat proses pencarian data.

Sistem basis data memiliki banyak model, salah satunya adalah basis data relasional. Basis data relasional menggunakan sekumpulan tabel dua dimensi yang masing-masing tabel tersusun atas baris dan kolom. Untuk membuat hubungan antara dua atau lebih tabel, digunakan *key* yaitu *primary key* di salah satu tabel dan *foreign key* di tabel lain.

Tabel 2.1, Tabel 2.2, dan Tabel 2.3 merupakan contoh tabel yang saling berhubungan. Kolom MHS_ID pada Tabel 2.1 merupakan *primary key* dan kolom MK_ID pada Tabel 2.2 juga merupakan *primary key*. MHS_ID dan MK_ID yang terdapat pada Tabel 2.3 disebut *foreign key*. Hal inilah yang menyebabkan tabel-tabel tersebut memiliki hubungan atau relasi.

Tabel 2.1 Tabel Mahasiswa

MHS_ID	NRP	NAMA	ALAMAT
1	5114100118	Muhammad Rifatullah	Lampung
2	5114100114	Shafly Naufal Adyant	Gombong
3	5114100066	Vinsensia Sipriana Zega	Medan
4	5114100105	Ivaldy Putra Lifiari	Jakarta
5	5114100104	fatihah `ulya hakiem	Surabaya

Tabel 2.2 Tabel Mata Kuliah

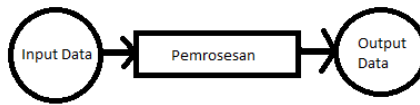
MK_ID	KODE_MK	NAMA_MK	SKS
1	BD012	Basis Data	4
2	DP001	Dasar Pemrograman	4
3	PW002	Pemrograman Web	3

Tabel 2.3 Tabel Nilai

MHS_ID	MK_ID	NILAI_ETS	NILAI_EAS
1	1	90	90
1	2	85	90
4	3	87	88
3	4	78	84

2.2 Sistem Informasi Manajemen

Sistem Informasi Manajemen (SIM) didefinisikan sebagai suatu alat untuk menyajikan informasi dengan cara sedemikian rupa sehingga bermanfaat bagi penerimanya [7]. Tujuannya adalah untuk menyajikan informasi guna pengambilan keputusan pada perencanaan, pemrakarsaan, pengorganisasian, pengendalian kegiatan operasi sub sistem suatu perusahaan, dan menyajikan sinergi organisasi pada proses [8]. Dengan demikian, sistem informasi berdasarkan konsep dapat dilihat pada Gambar 2.1.



Gambar 2.1 Konsep sistem informasi

2.3 Replikasi dan Sinkronisasi

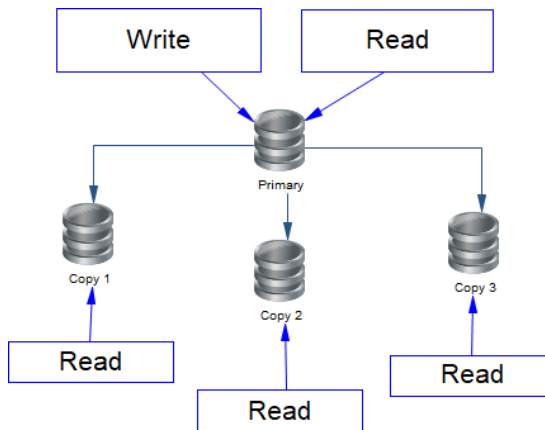
Replikasi adalah suatu teknik untuk melakukan *copy* dan pendistribusian data dari suatu basis data ke basis data lain, lalu dilakukan sinkronisasi antar basis data sehingga konsistensi data dapat terjamin. Sinkronisasi adalah suatu teknik untuk menyinkronkan data antara dua atau lebih basis data. Dengan menggunakan teknik replikasi, data dapat didistribusikan ke lokasi yang berbeda melalui jaringan lokal maupun internet. Replikasi juga memungkinkan untuk mendukung kinerja aplikasi, penyebaran data fisik sesuai dengan penggunaannya.

Dalam replikasi dan sinkronisasi, digunakan suatu aplikasi pihak ketiga untuk mencari atau lebih tepatnya mendeteksi perubahan yang terjadi suatu basis data. Setelah perubahan dalam satu basis data ter-identifikasi dan diketahui, maka perubahan akan diaplikasikan pada seluruh basis data replikanya.

Keuntungan replikasi tergantung dari jenis replikasi, tetapi pada umumnya replikasi mendukung ketersediaan (*availability*) data setiap waktu dan dimana pun diperlukan. Keuntungan lain adalah memungkinkan beberapa lokasi menyimpan data yang sama. Hal ini sangat berguna pada saat lokasi-lokasi tersebut membutuhkan data yang sama atau memerlukan *server* yang terpisah dalam pembuatan aplikasi.

2.3.1 *Primary Copy Replication*

Primary copy replication [2] merupakan salah satu metode replikasi. Pada teknik tersebut tabel pada basis data referensi direplikasi ke basis data yang membutuhkan dengan hak akses *read-only*. Semua manipulasi data diarahkan dan dieksekusi pada basis data referensi, kecuali *query select*. Aplikasi tidak melakukan *query select* ke basis data referensi, melainkan dilakukan pada basis data lokal. Metode ini meningkatkan performa sistem karena beban basis data referensi terbagi pada basis data lokal. Gambar 2.2 merupakan ilustrasi *primary copy replication*.



Gambar 2.2 Ilustrasi *Primary Copy Replication*

2.3.2 *Timestamp Based Database Synchronization*

Timestamp based database synchronization merupakan salah satu metode untuk sinkronisasi. Pada setiap tabel referensi terdapat kolom yang digunakan untuk mencatat waktu ketika terjadi perubahan data pada *record* tersebut. Kolom tersebut yang akan dibandingkan dengan waktu terakhir sinkronisasi (*last sync*) sehingga didapatkan data yang berubah saja, tidak semua data.

Beberapa istilah yang digunakan dalam metode *timestamp based database synchronization* adalah sebagai berikut.

1. *Last sync*

Merupakan waktu terakhir dilakukan sinkronisasi. *Last sync* disimpan pada suatu kolom yang disebut *last sync column*. *Last sync column* terletak pada aplikasi *synchronizer*.

2. *Time parameter*

Merupakan waktu ketika terjadi perubahan data. *Time parameter* disimpan pada suatu kolom yang disebut *time parameter column*. *Time parameter column* terletak pada setiap tabel referensi.

3. *Pulling*

Merupakan suatu proses penarikan data oleh aplikasi *synchronizer* pada tabel referensi dengan menarik data yang waktu pada *time parameter column* lebih besar atau sama dengan *last sync column*.

Tabel 2.4 merupakan contoh tabel pada aplikasi *synchronizer* yang berisi daftar nama tabel referensi dan waktu terakhir sinkronisasi (*last sync*) dengan *last sync column* adalah *last_sync*. Tabel 2.5 merupakan contoh tabel referensi. Kolom *updated_at* merupakan *time parameter column*. Hasil *pulling* pada tabel pengguna seperti pada Tabel 2.6. Hasil *pulling* inilah yang akan dikirimkan ke tabel replika.

Tabel 2.4 Contoh Tabel sync

id	nama_tabel	last_sync
1	pengguna	2017-02-06 13:06

Tabel 2.5 Contoh Tabel pengguna

id	nama	umur	alamat	updated_at
1	Dadang	19	Surabaya	2017-02-06 13:02
2	Rahmad	20	Magelang	2017-02-06 13:10
3	Rangga	21	Surabaya	2017-02-06 13:04
4	Ridwan	25	Klaten	2017-02-06 13:08
5	Yoga	22	Gresik	2017-02-06 13:06

Tabel 2.6 Hasil *Pulling* pada Tabel Pengguna

id	nama	umur	alamat	updated_at
2	Rahmad	20	Magelang	2017-02-06 13:10
4	Ridwan	25	Klaten	2017-02-06 13:08
5	Yoga	22	Gresik	2017-02-06 13:06

2.4 Publish/Subscribe Pattern

Publish/subscribe merupakan suatu paradigma pengiriman pesan dimana pihak yang memberikan informasi (*publishers*) tidak mempunyai hubungan langsung ke pihak yang menerima pesan (*subscribers*), namun interaksi antar-*publishers* dan *subscribers* dikontrol oleh *message broker* [3].

Karakteristik *publish/subscribe messaging* adalah setiap pesan dapat diterima oleh satu atau banyak *subscriber*. Selain itu, *publisher* dan *subscriber* memiliki ketergantungan waktu. Sebuah *Subscriber* yang berlangganan terhadap topik dapat mengonsumsi pesan yang baru diterbitkan setelah *subscriber* menjadi pelanggan.

2.4.1 *Message Queueing*

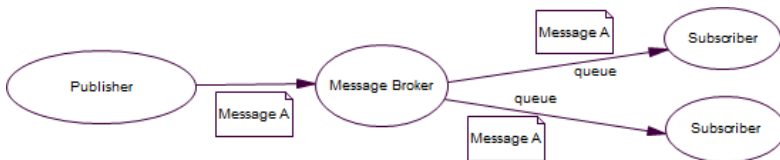
Saat ini konsep *message queueing* banyak digunakan pada sistem notifikasi. Dalam sistem *queue*, pesan akan disimpan pada node tertentu sampai sistem siap untuk meneruskannya. Misalnya, dalam kasus penyiaran (*broadcast*) notifikasi. Notifikasi tetap dikirimkan walaupun *user* belum berniat membukanya. Berdasarkan penjelasan tersebut, model *message queueing* memungkinkan pesan yang dikirimkan secara kontinu sekalipun salah satu server dalam keadaan tidak aktif. Pada tugas akhir ini akan digunakan RabbitMQ sebagai *message broker*. Gambar 2.3 menunjukkan cara kerja *message queueing* pada RabbitMQ. *Publisher* mengirimkan pesan ke *message broker*, lalu *message broker* akan mengirim pesan ke *queue*, kemudian pesan akan dikirimkan ke *subscriber*.

Beberapa istilah dalam RabbitMQ sebagai berikut.

1. *Message Broker*
Message Broker menerima pesan dari *publishers* dan meneruskan ke *subscriber*.
2. *Exchange*
Merupakan entitas dimana pesan dikirimkan. *Exchange* menerima pesan dan mengarahkan ke *queue*.
3. *Queue*
Tempat untuk menyimpan data yang bisa dikonsumsi oleh *subscriber*.
4. *Binding*
Aturan yang digunakan oleh *exchange* untuk meneruskan pesan ke *queue*. Agar *exchange* dapat mengirimkan pesan ke *queue*, *queue* harus di-*binding* ke *exchange*.

Analogi sederhana untuk memahami kaitan *exchange*, *queue*, dan *binding* adalah:

1. *Exchange* adalah Bandara Juanda.
2. *Queue* adalah tujuan, misal Keputih.
3. *Binding* adalah rute dari Bandara Juanda ke Keputih. Ada banyak cara untuk menuju Keputih.



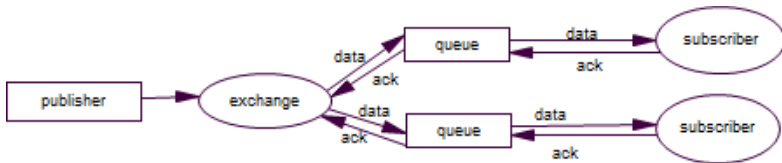
Gambar 2.3 Publish/Subscribe Model

2.4.2 Work Queue

Work queue (Task Queue) [9] adalah antrean suatu pekerjaan. *Work queue* akan digunakan untuk mendistribusikan tugas yang membutuhkan banyak sumber daya (*resource-consuming task*) di antara banyak *worker*. Tujuan utama pembuatan *work queue* adalah tidak mengerjakan *resource-consuming task* secara langsung dan harus menunggu sampai tugas tersebut selesai. Tugas akan di enkapsulasi sebagai suatu pesan dan dikirimkan ke suatu *queue*. Proses suatu *worker* yang berjalan di balik layar akan mengeksekusi tugas tersebut. Ketika banyak *worker* dijalankan tugas-tugas tersebut akan dibagi ke banyak *worker*. Salah satu keuntungan menggunakan *work queue* adalah kemampuan untuk bekerja secara paralel.

Sebuah tugas bisa dieksekusi dalam waktu yang cukup lama. Bagaimana jika *consumer* memulai sebuah tugas yang eksekusinya membutuhkan waktu yang lama namun ternyata tugas tersebut tidak sukses tereksekusi? RabbitMQ mendukung fitur *message acknowledgments* (ack). Pesan ack akan dikirim oleh *consumer* ke RabbitMQ server untuk menginstruksikan bahwa suatu pesan sudah diterima dan berhasil diproses lalu RabbitMQ boleh menghapus pesan tersebut.

Jika *consumer* mati karena *channel*-nya mati atau koneksi nya terputus maka RabbitMQ akan membuat ulang suatu *queue*. Dengan metode tersebut dapat dijamin bahwa tidak ada pesan yang hilang, bahkan ketika *worker* tiba-tiba mati. Gambar 2.4 merupakan gambaran proses pengiriman pesan dan ack pada RabbitMQ.



Gambar 2.4 Proses Pengiriman Pesan dan Acknowledgments (ack)

BAB III

ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini dibahas analisis kebutuhan, proses bisnis dan desain dari perangkat lunak yang dibangun dalam tugas akhir ini. Bagian awal bab akan dibahas tentang analisis permasalahan dan kebutuhan pengguna. Berikutnya dibahas fungsionalitas sistem berdasarkan hasil analisis kebutuhan. Bagian berikutnya akan dibahas rancangan perangkat lunak yang ditujukan untuk memberikan gambaran tentang perangkat lunak yang dibuat.

3.1 Analisis

Pada sub bab berikut akan dijelaskan analisis pembuatan kerangka kerja sinkronisasi basis data relasional pada sistem informasi berbasis web di ITS. Analisis yang dilakukan meliputi analisis permasalahan, kebutuhan umum perangkat lunak, deskripsi umum sistem, arsitektur dan fungsional sistem.

3.1.1 Analisis Permasalahan

DPTSI merupakan direktorat yang mengelola sistem informasi di ITS. DPTSI mengelola aplikasi web maupun perangkat bergerak. Namun setiap aplikasi mempunyai basis data masing-masing. Pada basis data sistem informasi kepegawaian (simpeg) terdapat data pegawai dan jabatannya, lalu akan dibuat sistem baru dengan nama e-Perkantoran, sistem informasi tentang persuratan. e-Perkantoran membutuhkan data pegawai sebagai data pengguna nya, juga data jabatan untuk digunakan sebagai alur persuratan dan pembagian hak akses sesuai dengan Organisasi dan Tata Kerja (OTK) ITS yang berlaku. Saat ini, cara yang digunakan adalah mengeksport data pegawai pada simpeg, lalu diimpor ke data pengguna pada e-Perkantoran. Tabel pegawai pada simpeg disebut tabel referensi, tabel pengguna pada e-Perkantoran disebut tabel replika.

Dengan data yang terduplikasi ini, sangat mungkin terjadi kesalahan. Misal terjadi perubahan pada tabel pegawai pengembang harus mengubah data tersebut secara manual pada tabel pengguna e-Perkantoran. Jika perubahan data hanya satu, maka mudah untuk menyalin data tersebut. Namun jika perubahan datanya sangat banyak sehingga sulit untuk diamati, maka penyalinan data secara manual tidak efektif.

Ilustrasi permasalahan yang dialami DPTSI perihal konsistensi dan integritas data seperti pada Tabel 3.1, Tabel 3.2, Tabel 3.3 dan Tabel 3.4. t merupakan variabel yang menunjukkan nomor perubahan. Tabel 3.1 merupakan contoh data ketika tabel Pegawai simpeg direplika ke tabel Pengguna e-Perkantoran. Tabel 3.1 menunjukkan data pada tabel referensi sama dengan tabel replika. Terjadi penambahan data pada tabel Pegawai, lalu Administrator Simpeg memberitahukan Administrator e-Perkantoran bahwa telah terjadi penambahan data. Tabel 3.2 menunjukkan kondisi tabel setelah dilakukan penambahan data ($t=1$). Tabel 3.3 menunjukkan kondisi tabel setelah dilakukan penambahan data lagi ($t = 2$). Lalu terjadi perubahan OTK sehingga banyak data yang berubah. Tabel 3.4 menunjukkan kondisi tabel setelah terjadi perubahan OTK ($t=3$).

Dengan perkembangan teknologi informasi, kondisi-kondisi seperti ini dapat diselesaikan dengan memanfaatkan kerangka kerja tambahan yang dapat mendeteksi perubahan data pada basis data referensi untuk dikirimkan ke basis data replika. Dengan memanfaatkan metode *timestamp based database synchronization*, kerangka kerja ini dapat mendeteksi perubahan data pada tabel referensi dan mengirim perubahan pada tabel referensi ke RabbitMQ server untuk didistribusikan ke *listener*-nya.

Tabel 3.1 Data ke $t = 0$ untuk (a) Data Pegawai Simpeg. (b) Data Pengguna e-Perkantoran.

ID	NAMA	UNIT
1	Drs. Agung Hermawan	1
2	Sandra Dewi	2

(a)

ID	NAMA	UNIT
1	Drs. Agung Hermawan	1
2	Sandra Dewi	2

(b)

Tabel 3.2 Data ke $t=1$ untuk (a) Data Pegawai Simpeg. (b) Data Pengguna e-Perkantoran.

ID	NAMA	UNIT
1	Drs. Agung Hermawan	1
2	Sandra Dewi	2
3	Ir.Arif Rahman	2

(a)

ID	NAMA	UNIT
1	Drs. Agung Hermawan	1
2	Sandra Dewi	2
3	Ir. Arif Rahman	2

(b)

Tabel 3.3 Data ke $t = 2$ untuk (a) Data Pegawai Simpeg. (b) Data Pengguna e-Perkantoran.

ID	NAMA	UNIT
1	Drs. Agung Hermawan	1
2	Sandra Dewi	2
3	Ir. Arif Rahman	2
4	Drs. Marsudi, M.S.	3
5	Drs. Edy Subali, M.Pd	4
6	Dra. Dyah Satya Yoga Agustin	3

(a)

ID	NAMA	UNIT
1	Drs. Agung Hermawan	1
2	Sandra Dewi	2
3	Ir. Arif Rahman	2
4	Ir. Edy Subali, M.Pd	3
5	Drs. Marsudi	4

(b)

Tabel 3.4 Data ke t = 3 untuk (a) Data Pegawai Simpeg. (b) Data Pengguna e-Perkantoran.

ID	NAMA	UNIT
1	Drs. Agung Hermawan	1
2	Sandra Dewi	2
3	Ir.Arif Rahman	1
4	Drs. Marsudi, M.S.	3
5	Drs. Edy Subali, M.Pd	2
6	Dra. Dyah Satya Yoga Agustin	2

(a)

ID	NAMA	UNIT
1	Drs. Agung Hermawan	1
2	Sandra Dewi	2
3	Ir. Arif Rahman	2
4	Ir. Edy Subali, M.Pd	3
5	Drs. Marsudi	4
6	Dra. Dyah Satya Yoga Agustina	3

(b)

3.1.2 Deskripsi Umum Perangkat Lunak

Kerangka kerja yang dibangun pada tugas akhir ini bernama Sync ITS. Sync ITS adalah kerangka kerja yang dibangun di atas kerangka kerja Laravel dengan bahasa pemrograman PHP. Sync ITS menggunakan metode *timestamp based database synchronization* dengan paradigma *publish/subscribe* untuk melakukan sinkronisasi.

Syarat-syarat implementasi Sync ITS adalah sebagai berikut.

1. Setiap tabel harus mempunyai *primary key*.
2. Tabel referensi dan tabel replika harus mempunyai tipe data yang sama.
3. Penghapusan data menggunakan *soft delete*.
4. *Primary Key* pada tabel replika tidak boleh *identity*.

Sync ITS mendeteksi perubahan dengan membandingkan waktu *last sync* dengan *time parameter column* pada tabel referensi. Setelah data ditarik (*pulling*), data akan dikirim ke *listener (subscriber)* dan akan diproses oleh *listener*.

Pertama, sistem akan membuat prioritas antrean. Hal ini bertujuan untuk mengurutkan tabel sesuai dependensinya pada setiap basis data. Proses ini penting karena semisal tabel RER.A berelasi ke tabel REF.B, dan keduanya harus disinkronisasi ke tabel REP.A dan REP.B, maka REF.B harus diproses terlebih dahulu baru REF.A. Jikalau terbalik, maka REF.A tidak akan bisa diproses dan besar kemungkinan akan terjadi *infinite loop*. Sebelum mengirim data, sistem akan mendeteksi apakah tabel referensi mengandung *self-reference relation*, yaitu relasi ke diri sendiri. Jika terdeteksi *self-reference*, maka data akan diproses dengan menggunakan algoritma *topological sorting*. *Topological sorting* merupakan suatu algoritma yang digunakan untuk mengurutkan sesuatu berdasarkan dependensinya. Hasil dari *topological sort* ini akan dipublish ke *listener*.

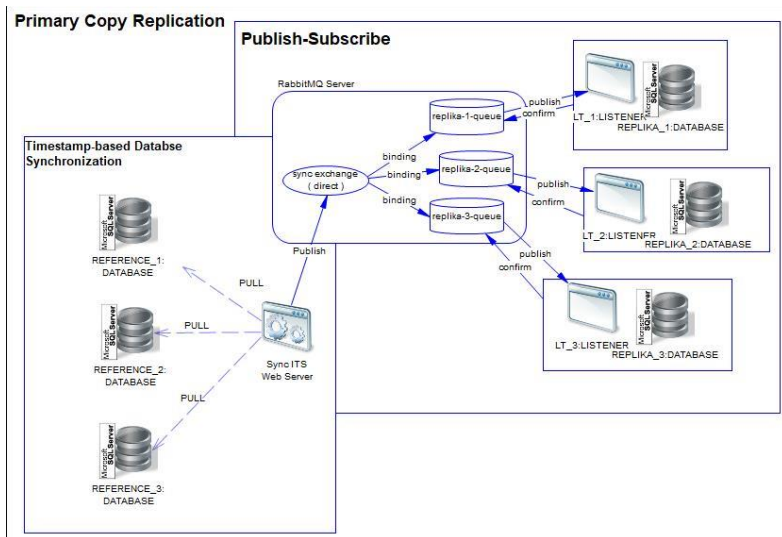
Setelah *listener* menerima data, langkah selanjutnya adalah memproses data. *Listener* akan mengecek apakah data yang dikirimkan sudah ada atau belum. Kalau belum, akan dilakukan penambahan data. Kalau sudah, akan dilakukan pembaruan data. Kalau *queue* gagal diproses, maka *queue* akan ditolak dan dipublish ulang sampai mencapai maksimal pengulangan. Jikalau lebih dari maksimal pengulangan, maka akan ditolak dan tidak dipublish ulang, serta Sync ITS akan mencatat pesan kesalahan yang terjadi.

Fungsionalitas Sync ITS apabila diringkas adalah sebagai berikut:

1. Sync ITS dapat membuat, mengubah, dan menghapus pengaturan koneksi ke basis data.

2. Sync ITS dapat membuat, mengubah, dan menghapus pengaturan *queue* antara tabel referensi dan replikanya.
3. Sync ITS dapat mendeteksi perubahan yang terjadi pada tabel referensi dan mengirimkannya ke tabel replika.
4. Sync ITS dapat melakukan pencatatan apabila ada *queue* yang gagal diproses oleh *listener*.

3.1.3 Arsitektur Sistem



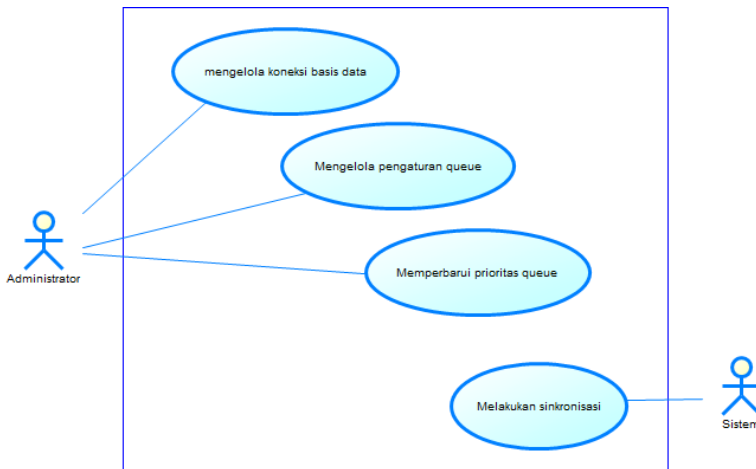
Gambar 3.1 Arsitektur Sistem

Kerangka kerja Sync ITS menggunakan arsitektur seperti pada Gambar 3.1. Sync ITS berperan sebagai pusat. Sync ITS bertanggung jawab untuk mengambil data dari tabel referensi, lalu mempublish data ke RabbitMQ Server pada *sync exchange* dengan *routing key* tertentu. *Sync exchange* akan mengarahkan pesan yang

dipublish oleh Sync ITS ke *subscriber* sesuai dengan *routing key*. *Subscriber* pada Sync ITS ini disebut *listener*. *Listener* akan memproses data. Jika sukses, *listener* akan mengkonfirmasi dengan membalas pesan *ack*. Jikalau gagal atau pesan tidak sampai pada *listener*, maka *queue* akan dipublish ulang.

3.1.4 Kebutuhan Fungsional Aplikasi

Kebutuhan fungsional kerangka kerja Sync ITS digambarkan pada Gambar 3.2.



Gambar 3.2 Diagram Kasus Penggunaan Sync ITS

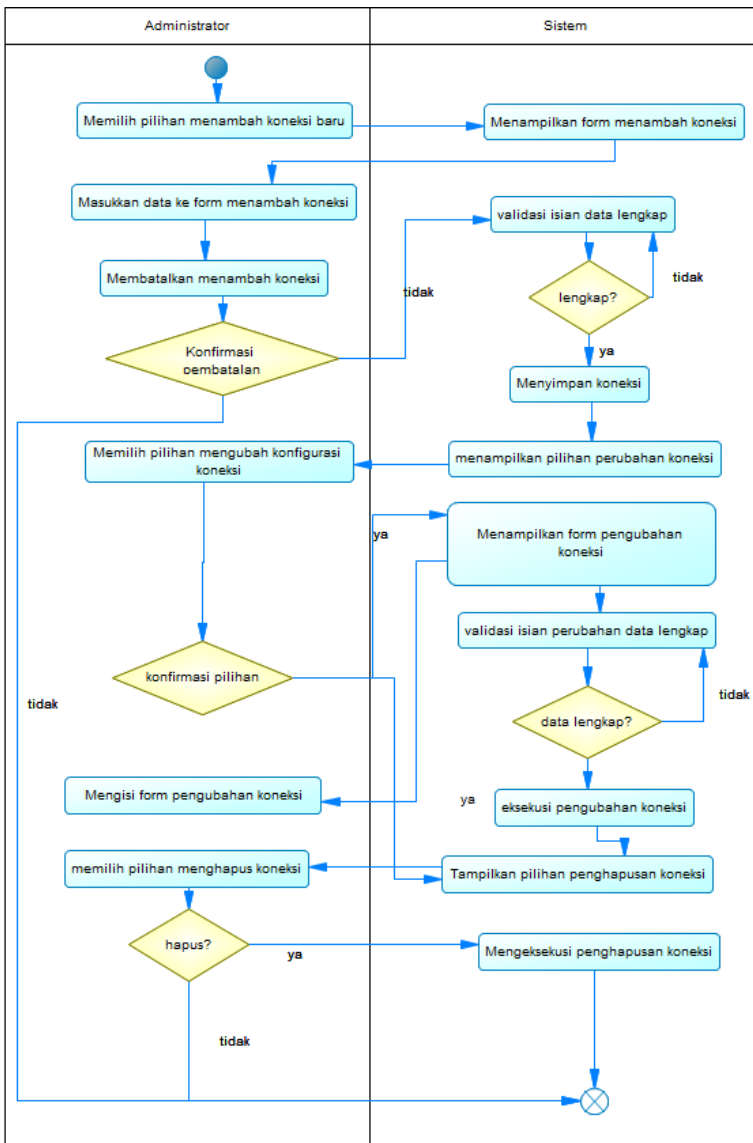
Hak akses penggunaan kerangka kerja Sync ITS hanya diberikan kepada Administrator basis data. Administrator dapat mengelola koneksi basis data, mengelola pengaturan *queue*, dan memperbarui prioritas *queue*. Penjelasan lengkap mengenai kasus penggunaan pada kerangka kerja Sync ITS berada pada Tabel 3.5.

Tabel 3.5 Deskripsi Kasus Penggunaan Sync ITS

No.	Kode	Nama	Keterangan
1	UC-001	Mengelola koneksi basis data	Administrator dapat mengelola koneksi basis data yaitu menambah, mengubah serta menghapus data
2	UC-002	Mengelola pengaturan <i>queue</i>	Administrator dapat mengelola pengaturan <i>queue</i> seperti menambah, mengubah serta menghapus data.
3	UC-003	Memperbarui prioritas <i>queue</i>	Administrator dapat memperbarui prioritas <i>queue</i> jikalau terjadi perubahan desain basis data.
4	UC-004	Melakukan sinkronisasi	Sistem melakukan sinkronisasi

3.1.4.1 Deskripsi Kasus Kebutuhan UC-001

Kasus kebutuhan kode UC-001 merupakan kasus kebutuhan mengelola koneksi basis data. Rincian alur kasus mengelola koneksi basis data dijelaskan pada Tabel 3.6 dan Gambar 3.3.



Gambar 3.3 Diagram Kasus Penggunaan UC-001

Tabel 3.6 Rincian Alur Kasus Penggunaan UC-001

Nama	Mengelola koneksi basis data
Nomor	UC-001
Aktor	Administrator
Kondisi Awal	Data pengaturan belum terisi dan <i>form</i> isian belum tampil
Kondisi Akhir	Informasi data koneksi tersimpan
Alur Normal	<ol style="list-style-type: none"> 1. Administrator memilih pilihan menambah koneksi baru 2. Sistem menampilkan form menambah koneksi 3. Administrator memasukkan data ke dalam <i>form</i> menambah koneksi <ol style="list-style-type: none"> A1. Administrator membatalkan menambah koneksi 4. Sistem memvalidasi isian data koneksi lengkap <ol style="list-style-type: none"> A2. Sistem mendeteksi isian tidak lengkap 5. Sistem menyimpan koneksi 6. Sistem menampilkan pilihan pengubahan konfigurasi koneksi <ol style="list-style-type: none"> A3. Administrator tidak memilih pilihan pengubahan koneksi 7. Administrator memilih pilihan mengubah konfigurasi koneksi 8. Sistem menampilkan <i>form</i> pengubahan koneksi 9. Administrator mengisi <i>form</i> pengubahan koneksi <ol style="list-style-type: none"> A4. Administrator membatalkan proses pengubahan konfigurasi 10. Sistem memvalidasi isian data lengkap <ol style="list-style-type: none"> A5. Sistem mendeteksi isian data konfigurasi koneksi tidak lengkap

Tabel 3.7 Rincian Alur Kasus Penggunaan UC-001 (lanjutan)

Alur Normal (lanjutan)	11. Sistem mengeksekusi pengubahan koneksi 12. Sistem menampilkan pilihan penghapusan koneksi. A6. Administrator tidak memilih pilihan penghapusan koneksi. 13. Administrator memilih pilihan penghapusan koneksi. A7. Administrator membatalkan proses penghapusan koneksi. 14. Sistem mengeksekusi penghapusan koneksi. 15. Kasus penggunaan mengelola koneksi basis data berakhir.
	A1. Administrator membatalkan menambah koneksi. A1.1 Menuju alur normal nomor 15. A2. Sistem mendeteksi isian tidak lengkap. A2.1 Menuju alur normal nomor 2. A3. Administrator tidak memilih pilihan pengubahan koneksi. A3.1 Menuju alur normal nomor 12. A4. Administrator membatalkan proses pengubahan konfigurasi A4.1 Menuju alur normal nomor 6. A5. Sistem mendeteksi isian data konfigurasi koneksi tidak lengkap A5.1 Menuju alur nomor 8. A6. Administrator tidak memilih pilihan penghapusan koneksi. A6.1 Menuju alur normal nomor 15. A7. Administrator membatalkan proses penghapusan koneksi. A7.1 Menuju alur normal nomor 12.

3.1.4.2 Deskripsi Kasus Kebutuhan UC-002

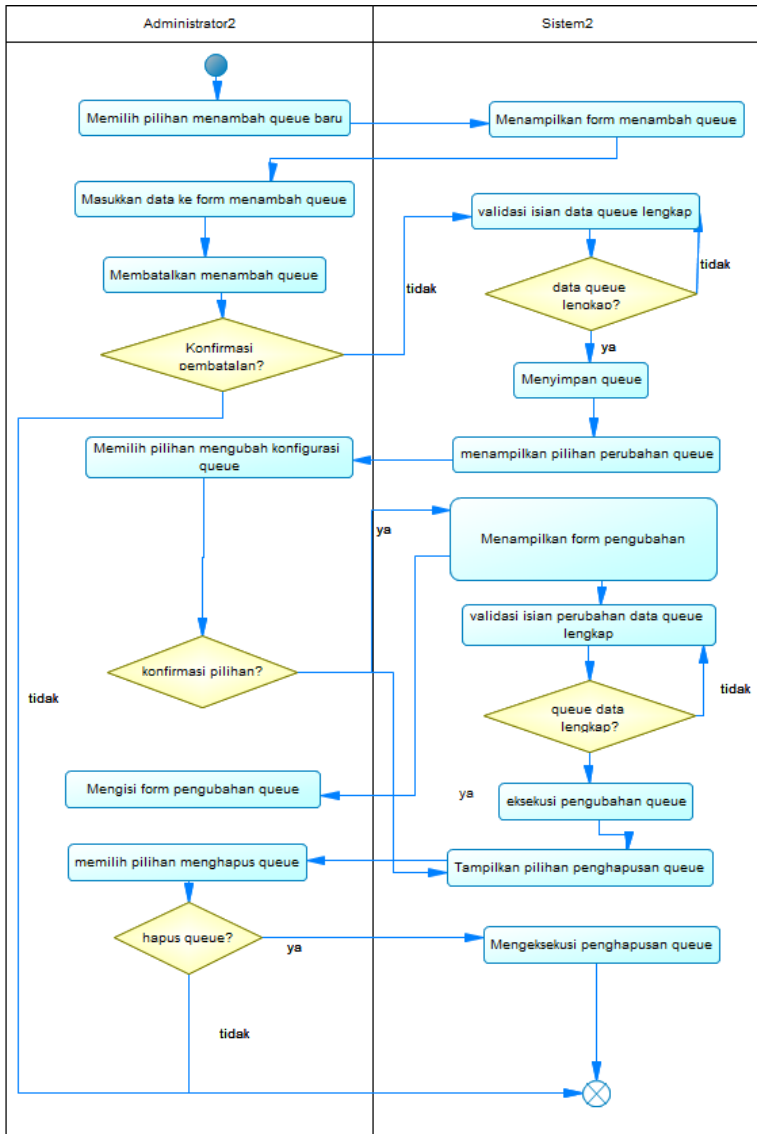
Kasus kebutuhan kode UC-002 merupakan kasus kebutuhan mengelola *queue*. Rincian alur kasus mengelola *queue* dijelaskan pada Gambar 3.4 dan Tabel 3.8.

Tabel 3.8 Rincian Alur Kasus Penggunaan UC-002

Nama	Mengelola pengaturan <i>queue</i>
Nomor	UC-002
Aktor	Administrator
Kondisi Awal	Data pengaturan belum terisi dan <i>form</i> isian belum tampil
Kondisi Akhir	Informasi data <i>queue</i> tersimpan
Alur Normal	<ol style="list-style-type: none"> 1. Administrator memilih pilihan menambah <i>queue</i> baru 2. Sistem menampilkan form menambah <i>queue</i> 3. Administrator memasukkan data ke dalam <i>form</i> menambah <i>queue</i> <ol style="list-style-type: none"> A1. Administrator membatalkan menambah <i>queue</i> 4. Sistem memvalidasi isian data <i>queue</i> lengkap <ol style="list-style-type: none"> A2. Sistem mendeteksi isian tidak lengkap 5. Sistem menyimpan <i>queue</i> 6. Sistem menampilkan pilihan pengubahan konfigurasi <i>queue</i> <ol style="list-style-type: none"> A3. Administrator tidak memilih pilihan pengubahan <i>queue</i> 7. Administrator memilih pilihan mengubah konfigurasi <i>queue</i> 8. Sistem menampilkan <i>form</i> pengubahan <i>queue</i> 9. Administrator mengisi <i>form</i> pengubahan <i>queue</i> <ol style="list-style-type: none"> A4. Administrator membatalkan proses pengubahan konfigurasi

Tabel 3.9 Rincian Alur Kasus Penggunaan UC-002 (lanjutan)

	<p>10. Sistem memvalidasi isian data lengkap A5. Sistem mendeteksi isian data <i>queue</i> tidak lengkap</p> <p>11. Sistem mengeksekusi pengubahan <i>queue</i></p> <p>12. Sistem menampilkan pilihan hapus <i>queue</i>. A6. Administrator tidak memilih pilihan hapus <i>queue</i>.</p> <p>13. Administrator memilih pilihan hapus <i>queue</i>. A7. Administrator membatalkan proses penghapusan <i>queue</i>.</p> <p>14. Sistem mengeksekusi penghapusan <i>queue</i>.</p> <p>15. Kasus penggunaan mengelola <i>queue</i> basis data berakhir</p>
	<p>A1. Administrator membatalkan menambah <i>queue</i>. A1.1 Menuju alur normal nomor 15.</p> <p>A2. Sistem mendeteksi isian tidak lengkap. A2.1 Menuju alur normal nomor 2.</p> <p>A3. Administrator tidak memilih pilihan pengubahan <i>queue</i>. A3.1 Menuju alur normal nomor 12.</p> <p>A4. Administrator membatalkan proses pengubahan pengaturan <i>queue</i> A4.1 Menuju alur normal nomor 6.</p> <p>A5. Sistem mendeteksi isian data pengaturan <i>queue</i> tidak lengkap A5.1 Menuju alur nomor 8.</p> <p>A6. Administrator tidak memilih pilihan penghapusan <i>queue</i>. A6.1 Menuju alur normal nomor 15.</p> <p>A7. Administrator membatalkan proses penghapusan <i>queue</i>. A7.1 Menuju alur normal nomor 12.</p>



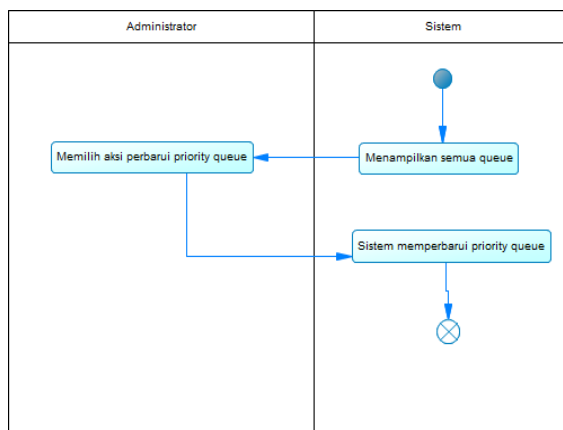
Gambar 3.4 Diagram Kasus Penggunaan UC-002

3.1.4.3 Deskripsi Kasus kebutuhan UC-003

Kasus kebutuhan kode UC-003 merupakan kasus kebutuhan memperbarui prioritas *queue*. Rincian alur kasus memperbarui prioritas *queue* dijelaskan pada Gambar 3.5 dan Tabel 3.10.

Tabel 3.10 Rincian Alur Kasus Penggunaan UC-003

Nama	Memperbarui prioritas <i>queue</i>
Nomor	UC-003
Aktor	Administrator
Kondisi Awal	Urutan <i>queue</i> pada basis data belum diperbaharui.
Kondisi Akhir	Urutan <i>queue</i> pada basis data sudah diperbaharui.
Alur Normal	<ol style="list-style-type: none"> 1. Sistem menampilkan semua <i>queue</i> 2. Administrator memilih aksi perbarui prioritas <i>queue</i> 3. Sistem memperbarui prioritas <i>queue</i>. 4. Kasus penggunaan memperbarui prioritas <i>queue</i> berakhir.



Gambar 3.5 Diagram Kasus Penggunaan UC-003

3.1.4.4 Deskripsi Kasus Kebutuhan UC-004

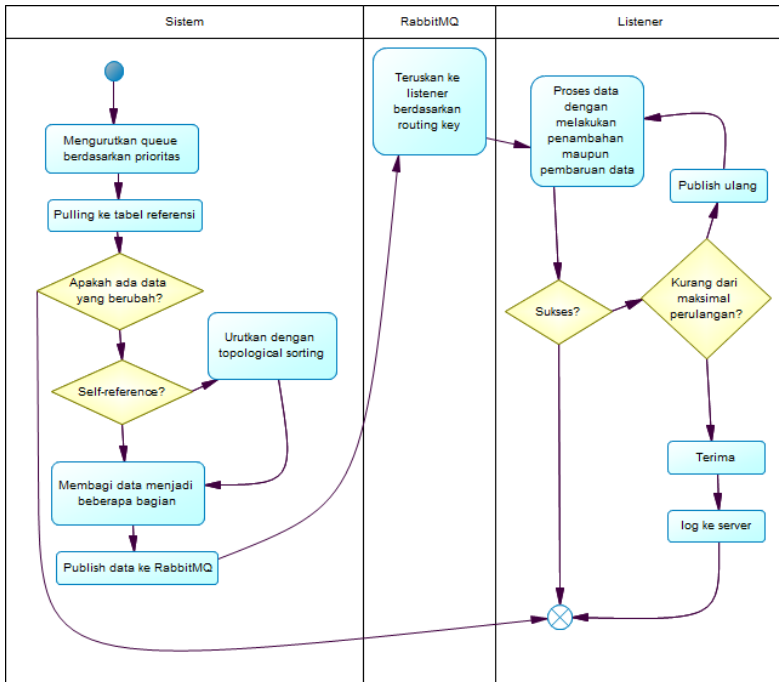
Kasus kebutuhan kode UC-004 merupakan kasus kebutuhan sinkronisasi. Rincian alur kasus penggunaan UC-004 seperti pada Tabel 3.11 dan Gambar 3.6.

Tabel 3.11 Rincian Alur Kasus Penggunaan UC-004

Nama	Melakukan sinkronisasi
Nomor	UC-004
Aktor	Sistem
Kondisi Awal	Tabel referensi tidak sama dengan tabel replika
Kondisi Akhir	Tabel referensi sama dengan tabel replika
Alur Normal	<ol style="list-style-type: none"> 1. Sistem mengurutkan <i>queue</i> berdasarkan prioritas 2. Sistem melakukan <i>pulling</i> ke tabel referensi A2. Tidak ada data yang dikirimkan 3. Sistem mendeteksi apakah terdapat <i>self-reference</i> <ol style="list-style-type: none"> A1. Sistem mendeteksi <i>self-reference</i> 4. Sistem membagi data menjadi beberapa bagian 5. Sistem mempublish data ke RabbitMQ 6. RabbitMQ mengirimkan data ke <i>listener</i>. 7. <i>Listener</i> memproses data <ol style="list-style-type: none"> A3. Data gagal diproses 8. Sinkronisasi selesai.
	<ol style="list-style-type: none"> A1. Sistem mendeteksi <i>self-reference</i>. <ol style="list-style-type: none"> A1.1 Data diurutkan dengan menggunakan <i>topological sorting algorithm</i>. A1.2 Menuju alur nomor 4 A2. Tidak ada data yang dikirimkan <ol style="list-style-type: none"> A2.1 Menuju alur nomor 8 A3. Data gagal diproses <ol style="list-style-type: none"> A3.1 Terima data dan publish ulang hingga maksimal perulangan. <ol style="list-style-type: none"> A3.1.1 Melebihi maksimal perulangan A3.2 Menuju alur nomor 7.

Tabel 3.12 Rincian Alur Kasus Penggunaan UC-004 (lanjutan)

	A3.1.1 Melebihi maksimal perulangan A3.1.1.1 Terima dan jangan publish ulang A3.1.1.2 Kirim log ke server A3.1.1.3 Menuju alur nomor 8
--	---

**Gambar 3.6 Diagram Kasus Penggunaan UC-004**

3.2 Perancangan

3.2.1 Perancangan Proses

Pada sub bab ini akan dibahas secara mendetail dari rancangan proses Sync ITS untuk memenuhi kebutuhan fungsionalnya.

3.2.1.1 Proses Menambah Koneksi

Proses menambah koneksi merupakan proses yang dilakukan ketika Administrator akan membuat *queue* baru. Koneksi baru dibuat melalui antarmuka Sync ITS. Nama koneksi harus unik, tidak boleh ada yang sama. Satu koneksi berlaku untuk satu basis data, sedangkan satu koneksi bisa untuk banyak *queue*.

3.2.1.2 Proses Mengubah Koneksi

Proses mengubah koneksi merupakan proses yang dilakukan ketika informasi koneksi akan diubah. Proses perubahan koneksi dilakukan melalui antarmuka Sync ITS.

3.2.1.3 Proses Menghapus Koneksi

Proses menghapus koneksi merupakan proses yang dilakukan ketika pengaturan tidak lagi dibutuhkan. Proses penghapusan koneksi menggunakan *soft deletes*.

3.2.1.4 Proses Menambah Queue

Proses menambah *queue* merupakan proses yang dilakukan ketika Administrator akan membuat *queue* baru untuk mengirim data dari tabel referensi ke tabel replika. *Queue* baru dibuat melalui antarmuka Sync ITS. Pengaturan yang dibuat akan disimpan pada basis data Sync ITS. Data yang harus disimpan adalah nama *queue*, tabel referensi dan replikanya, *mapping* antara kolom tabel referensi dengan kolom tabel replika, serta waktu terakhir sinkronisasi dilakukan.

Proses menambah *queue* ini hanya menambah data pada tabel, bukan membuat *queue* pada RabbitMQ Server. Pembuatan

queue pada RabbitMQ Server terjadi setelah *listener* yang sudah diunduh dijalankan.

3.2.1.5 Proses Mengubah Queue

Proses mengubah *queue* merupakan proses yang dilakukan ketika informasi *queue* akan diubah. Seluruh isi pengaturan *queue* dapat diubah kecuali nama *queue*. Perubahan dilakukan melalui antarmuka Sync ITS.

3.2.1.6 Proses Menghapus Queue

Proses menghapus *queue* merupakan proses yang dilakukan ketika *queue* tidak lagi dibutuhkan. Proses menghapus *queue* dilakukan melalui antarmuka Sync ITS. Penghapusan *queue* menggunakan *soft delete*.

3.2.1.7 Proses Sinkronisasi Basis Data

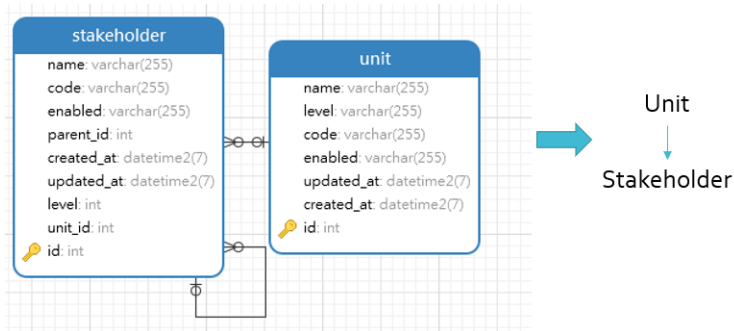
Proses sinkronisasi basis data merupakan proses yang dilakukan sesuai jadwal yang telah ditentukan. Sync ITS menggunakan fitur bawaan Laravel untuk penjadwalan [10] dengan menggunakan bantuan cronjob.

Proses sinkronisasi diurutkan berdasarkan prioritas. Prioritas suatu tabel ditentukan berdasarkan relasi tabel tersebut ke tabel lain. Tabel yang tidak berelasi ke tabel lain akan diproses terlebih dahulu. Gambar 3.7 merupakan contoh penentuan prioritas sinkronisasi. Tabel stakeholder berelasi ke tabel unit, maka tabel unit diproses terlebih dahulu.

Istilah-istilah dalam proses sinkronisasi adalah sebagai berikut.

1. *Self-referencing*

Tabel yang berelasi terhadap dirinya sendiri. Tabel 3.13 merupakan contoh tabel *self-referencing*. Kolom *manager_id* berelasi terhadap kolom *id*.



Gambar 3.7 Contoh Prioritas Tabel

2. *Circular Dependency*

Circular dependency adalah relasi antar dua atau lebih data yang saling bergantung satu sama lain. Data dengan id 3 dan 4 pada tabel Tabel 3.13 merupakan contoh data yang *circular*. Id 3 berelasi dengan id 4, id 4 berelasi dengan id 3.

Tabel 3.13 Contoh table pengguna

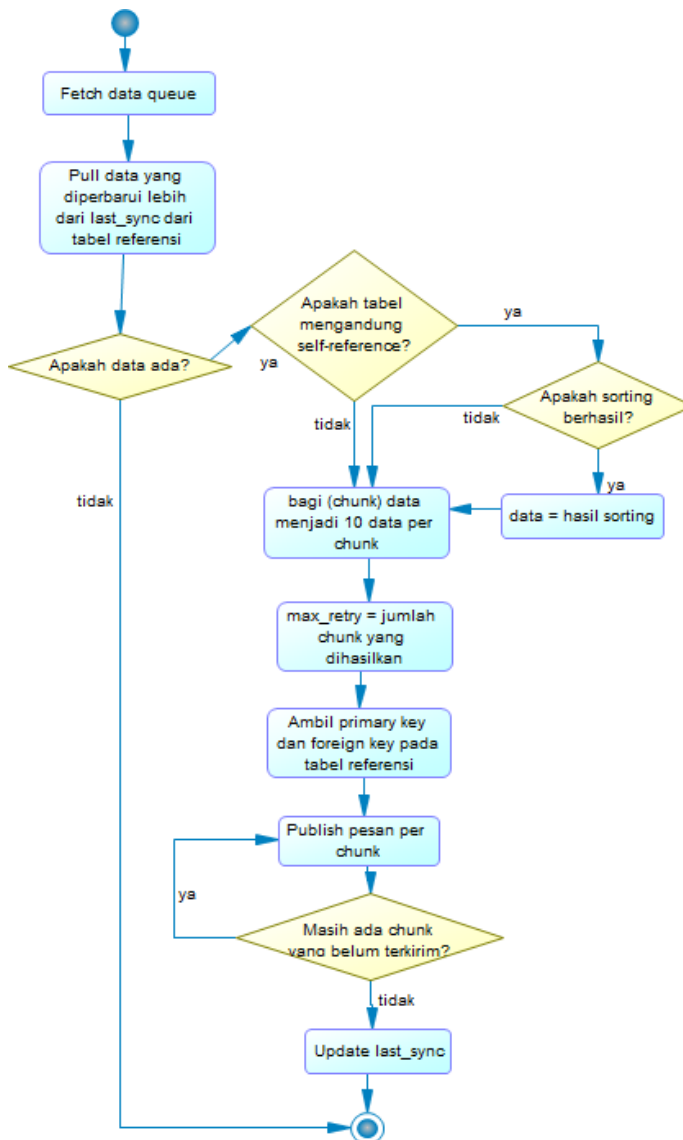
Id	Nama	Manager_id
1	A	Null
2	B	1
3	C	4
4	D	3

Secara umum, urutan proses sinkronisasi basis data adalah sebagai berikut.

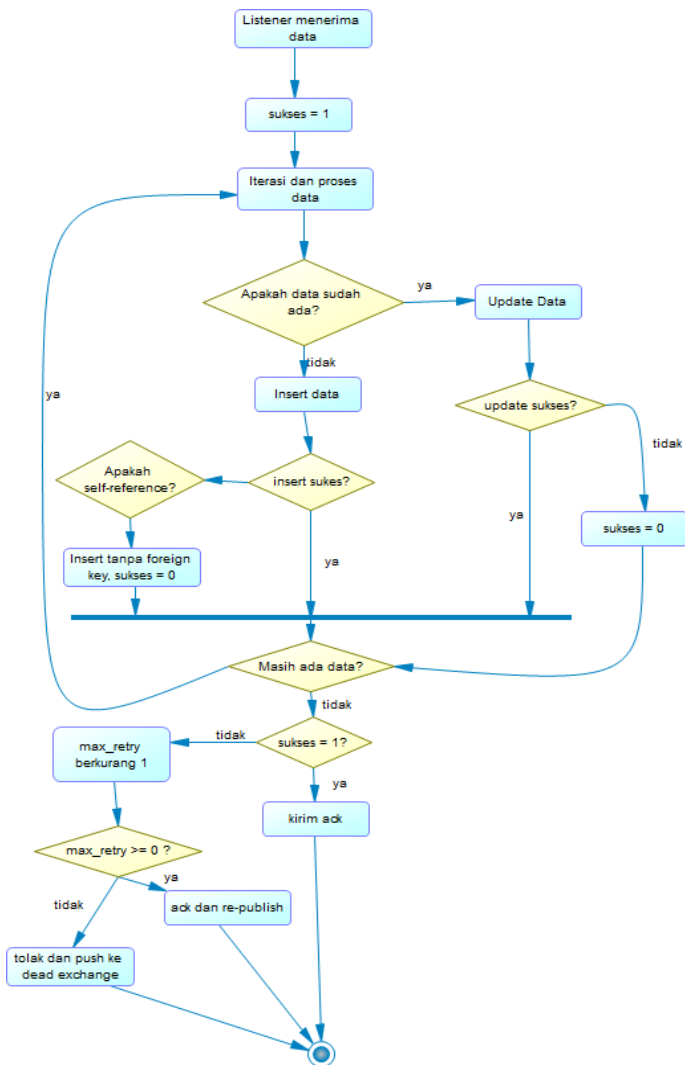
1. Asumsi proses sinkronisasi terjadi pada waktu T.
2. Sync ITS melakukan *pulling* ke tabel referensi.
3. Sebelum data dikirim, Sync ITS akan mengecek apakah tabel referensi mengandung *self-reference*. Jika tabel replikasi mengandung *self-reference*, maka data akan diurutkan dengan menggunakan algoritma *topological sorting*.

4. Data dibagi menjadi beberapa bagian (*chunk*) lalu setiap bagian dipublish ke RabbitMQ pada *exchange* dengan nama “sync” dan *routing key* sesuai nama *queue*-nya.
5. Setelah data dipublish, perbarui *last sync column* dengan waktu T seperti pada nomor 1.
6. *Listener* menerima data, lalu memproses data.
7. Jika data tidak ditemukan, maka tambah data baru. Jika data ditemukan tapi nilai hash-nya sama, maka lewati. Jika data ditemukan dan nilai hash-nya berbeda, maka perbarui data. Nilai hash dari suatu *record* tersimpan pada Redis.
8. Apabila terdeteksi *circular*, maka data ditambahkan ke tabel replika tanpa nilai *foreign key* nya. *Listener* akan mengirimkan pesan *ack* dan akan mempublish ulang *queue* tersebut. Kembali ke nomor 5.
9. Setiap *queue* mempunyai maksimal perulangan. Jika melebihi maksimal perulangan, maka *queue* akan ditolak dan tidak akan dipublish ulang.
10. Untuk kondisi *queue* ditolak dan tidak dipublish ulang (*dead letter queue*), *queue* akan dikirim ke *dead letter exchange* dengan nama *sync.dead* lalu *listener* akan mengirimkan perintah kepada Sync ITS untuk mencatat log yang berisi *queue_list_id*, isi *queue message* dan *error message*.

Queue tidak akan di-*drop* apabila tidak ada *ack* dari *listener*. Skema ini bermanfaat jikalau koneksi *listener* tiba-tiba putus atau terjadi kesalahan kode. Ketika *listener* tersambung lagi, maka secara otomatis RabbitMQ akan mempublish *queue* yang belum terkirim atau *queue* yang belum di-*ack*. Gambar 3.8 menggambarkan Sync ITS mempublish data, sedangkan Gambar 3.9 menggambarkan *listener* memproses data yang diterima.



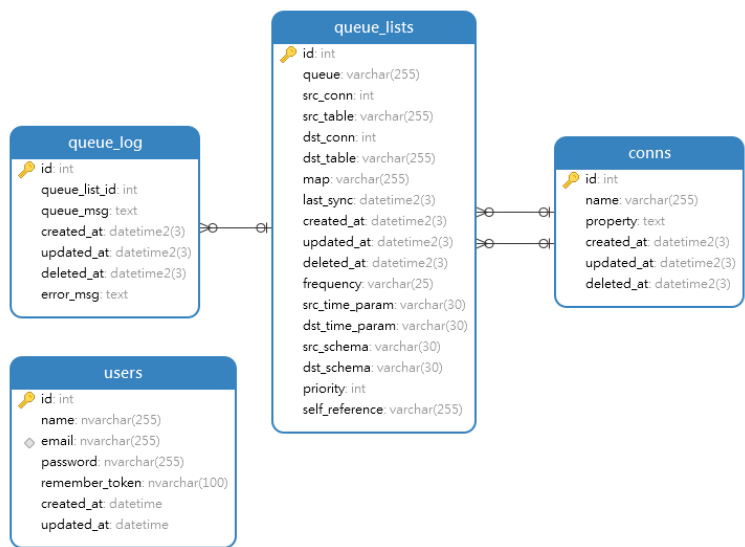
Gambar 3.8 Diagram Alur Sync ITS Publish Data



Gambar 3.9 Diagram Alur *Listener* Memproses Data

3.2.2 Perancangan Basis Data

Pada sub bab ini akan dijelaskan bagaimana rancangan basis data yang digunakan pada perangkat kerja Sync ITS sesuai perancangan pada sub bab 3.2.1. Basis data yang digunakan pada sistem yang dibangun pada tugas akhir ini menggunakan sistem manajemen basis data relasional SQL Server. SQL Server digunakan untuk menyimpan keterangan pengaturan koneksi basis data, data *queue* beserta tabel referensi dan replikanya, serta log *queue* yang ditolak. *Physical Data Model* (PDM) dari basis data sistem ini dapat dilihat pada Gambar 3.10.



Gambar 3.10 PDM Sync ITS

3.2.2.1 Rancangan Tabel conns

Tabel *conns* digunakan untuk menyimpan koneksi basis data yang ada di ITS. Detail atribut tabel *conns* dijelaskan pada Tabel 3.14.

Tabel 3.14 Atribut tabel conns

Nama Atribut	Tipe Data	Keterangan
id	Integer	<i>Primary Key</i> tabel conns.
name	Varchar	Nama koneksi
property	Text	Data konfigurasi untuk membuat koneksi ke basis data seperti <i>driver</i> , <i>host</i> , <i>port</i> , <i>database</i> , <i>username</i> , dan <i>password</i> . Password yang tersimpan pada tabel queue_log terenkripsi.
created_at	Datetime2	Waktu pembuatan data
updated_at	Datetime2	Waktu perubahan data terjadi
deleted_at	Datetime2	Waktu data dihapus (<i>soft delete</i>)

3.2.2.2 Rancangan Tabel queue_lists

Tabel queue_lists digunakan untuk menyimpan daftar tabel referensi dan replikanya, serta nama *queue* yang akan digunakan sebagai parameter *queue name* dan *routing key* pada saat pembuatan *queue* oleh *listener*. Atribut pada tabel queue_lists seperti pada Tabel 3.15.

Tabel 3.15 Atribut pada tabel queue_lists

Nama Atribut	Tipe Data	Keterangan
id	Integer	<i>Primary key</i> dari tabel queue_lists
queue	Varchar (255)	Nama <i>queue</i>
src_conn	Integer	Koneksi ke basis data referensi
src_schema	Varchar (30)	Nama schema tabel referensi
src_tabel	Varchar (255)	Nama tabel referensi
src_time_param	Varchar (30)	Nama kolom pencatat waktu perubahan data pada tabel referensi (<i>time parameter column</i>)
dst_conn	Integer	Koneksi ke basis data replika
dst_schema	Varchar (30)	Nama schema tabel replika
dst_tabel	Varchar (255)	Nama tabel replika
dst_time_param	Varchar (30)	Nama kolom pencatat waktu perubahan data pada tabel replika (<i>time parameter column</i>)
map	Varchar (255)	<i>Mapping</i> antara nama kolom tabel referensi dan tabel replika
frequency	Varchar (25)	Penjadwalan dalam satuan menit
priority	Integer	Prioritas <i>queue</i>
self_reference	Varchar (255)	Data <i>foreign key</i> dan <i>primary key</i> pada tabel referensi (Terisi jika pada tabel terdeteksi relasi ke tabel nya sendiri)

last_sync	Datetime2	Waktu terakhir sinkronisasi (<i>last sync column</i>)
created_at	Datetime2	Waktu pembuatan data
updated_at	Datetime2	Waktu perubahan data terjadi
deleted_at	Datetime2	Waktu data dihapus (<i>soft delete</i>)

3.2.2.3 Rancangan Tabel queue_log

Pada saat *listener* gagal memproses data, *queue* akan dipublish ulang sampai *n* kali. Nilai *n* tergantung dari total data yang dikirim. Ketika *queue* sudah dipublish ulang sampai *n* kali namun tetap terjadi *error*, maka *queue* akan ditolak dan tidak akan dikirimkan kembali. Tabel *queue_log* digunakan untuk mencatat *error log* pada *queue* yang ditolak tersebut sehingga admin bisa mengetahui jikalau terjadi *error* dan mengambil tindakan yang diperlukan. Atribut pada tabel *queue_log* seperti pada Tabel 3.16.

Tabel 3.16 Atribut Pada Tabel queue_log

Nama Atribut	Tipe Data	Keterangan
id	Integer	<i>Primary Key</i> tabel <i>queue_log</i>
queue_list_id	Integer	<i>Foreign Key</i> ke tabel <i>queue_lists</i>
queue_msg	Text	Berisi pesan pada suatu <i>queue</i> yang gagal
error_msg	Text	<i>Error message</i> ketika <i>queue</i> di- <i>reject</i> dan melebihi maksimal pengulangan
created_at	Datetime2	Waktu pembuatan data
updated_at	Datetime2	Waktu perubahan data terjadi
deleted_at	Datetime2	Waktu data dihapus (<i>soft delete</i>)

3.2.2.4 Rancangan Tabel users

Tabel users digunakan untuk menyimpan data pengguna. Atribut pada tabel users seperti pada Tabel 3.17.

Tabel 3.17 Atribut Pada Tabel users

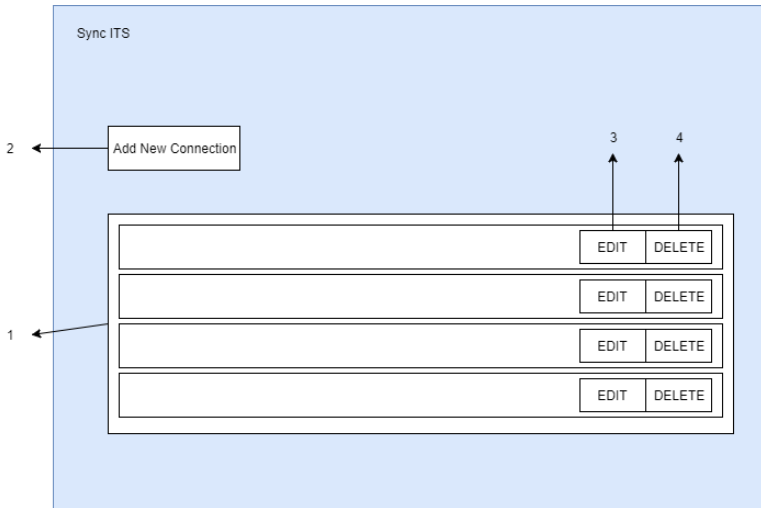
Nama Atribut	Tipe Data	Keterangan
id	Integer	<i>Primary Key</i> tabel users
name	Varchar (255)	Nama pengguna
email	Varchar (255)	Email pengguna
password	Varchar (255)	Password pengguna
remember_token	Varchar (100)	Token ketika pengguna memilih “remember me”
created_at	Datetime2	Waktu data dibuat
updated_at	Datetime2	Waktu data diubah

3.2.3 Perancangan Antarmuka

Pada sub bab ini akan dibahas secara detail perihal rancangan antarmuka kerangka kerja Sync ITS.

3.2.3.1 Antarmuka Halaman Menu Koneksi

Gambar 3.11 merupakan gambar rancangan antarmuka halaman menu koneksi. Halaman ini menampilkan daftar koneksi yang sudah dibuat beserta tombol aksi untuk mengelola konfigurasi.



Gambar 3.11 Rancangan Antarmuka Halaman Menu Koneksi

Berikut penjelasan masing-masing nomor yang tertera pada Gambar 3.11.

1. Daftar koneksi yang sudah pernah dibuat.
2. Tombol untuk memunculkan *modal* tambah koneksi data.
3. Tombol untuk memunculkan *modal* ubah koneksi.
4. Tombol untuk menghapus koneksi.

The image shows a 'Create New Connection' dialog box with the following fields and controls:

- Connection Name**: A text input field.
- Driver**: A dropdown menu (labeled 5).
- Host**: A text input field.
- Port**: A text input field.
- Database**: A text input field.
- Username**: A text input field.
- Password**: A text input field.
- Submit**: A button (labeled 2).
- Close**: A button (labeled 3).

Annotations and their targets:

- 1**: Points to the 'Port' field.
- 2**: Points to the 'Submit' button.
- 3**: Points to the 'Close' button.
- 4**: A cluster of arrows pointing to the 'Host', 'Port', 'Database', 'Username', and 'Password' fields.
- 5**: Points to the 'Driver' dropdown menu.

Gambar 3.12 Antarmuka Modal untuk Membuat koneksi baru

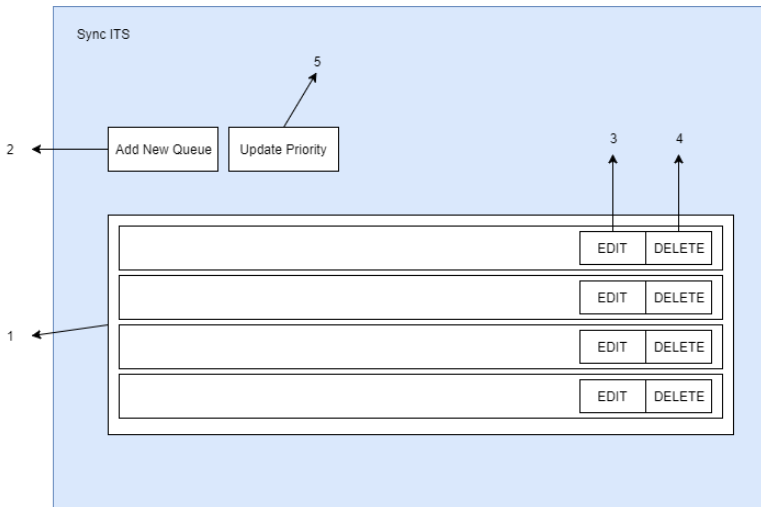
Gambar 3.12 merupakan desain antarmuka untuk menampilkan *form* penambahan atau perubahan data. Berikut penjelasan masing-masing nomor yang tertera pada Gambar 3.12.

1. Form untuk mengisi data koneksi baru
2. Tombol untuk menyimpan koneksi baru
3. Tombol untuk menutup modal membuat koneksi baru
4. *Textbox* untuk menampung input Administrator
5. *Combobox* untuk memilih *driver* yang tersedia

Antarmuka halaman mengubah koneksi sama dengan antarmuka halaman tambah koneksi baru (Gambar 3.12). Adapun yang membedakan adalah pada halaman mengubah koneksi data sudah terisi berdasarkan koneksi yang sudah disimpan sebelumnya.

3.2.3.2 Antarmuka Halaman Menu Daftar *Queue*

Gambar 3.13 merupakan gambar rancangan antarmuka halaman menu daftar *queue*. Pada halaman tersebut menampilkan daftar *queue* yang sudah dibuat beserta tombol aksi untuk mengelola *queue*.



Gambar 3.13 Rancangan Antarmuka Halaman Daftar *Queue*

Berikut penjelasan masing-masing nomor yang tertera pada Gambar 3.13.

1. Daftar *queue* yang sudah dibuat
2. Tombol menuju halaman tambah *queue*
3. Tombol untuk menuju halaman mengubah pengaturan *queue*.
4. Tombol untuk menghapus *queue*.
5. Tombol untuk memperbarui prioritas *queue*.

3.2.3.3 Antarmuka Halaman Tambah *Queue*

Gambar 3.13 merupakan gambar rancangan antarmuka untuk menambah *queue*. Pada halaman tersebut menampilkan *form* yang harus diisi untuk membuat suatu data *queue*.

The diagram illustrates the 'Add Queue' interface. It features a light blue background with a white form area. At the top, there are two side-by-side sections labeled 'Source' and 'Destination'. Each section contains three dropdown menus: 'Connection', 'Schema', and 'Table'. A line with the number '1' points to the 'Connection' dropdown in the 'Source' section. Below these sections is a table with two columns: 'Source Column' and 'Target Column'. The 'Target Column' column has three dropdown menus. A line with the number '2' points to the first row of the table, and a line with the number '3' points to the first dropdown menu in the 'Target Column' column. Below the table are three input fields: 'frequency', 'src_time_param', and 'dst_time_param'. A line with the number '4' points to the 'frequency' input field. A line with the number '5' points to the 'dst_time_param' dropdown menu. At the bottom right is a 'Submit' button, with a line and the number '6' pointing to it.

Gambar 3.14 Rancangan Antarmuka untuk Menambah *Queue*

Berikut penjelasan masing-masing nomor yang tertera pada Gambar 3.13.

1. *Combobox* untuk memilih koneksi, schema, dan tabel.
2. Daftar kolom pada tabel referensi.
3. Setiap *Combobox* yang berisi kolom pada tabel replika.
4. *Textbox* untuk menerima input frekuensi *queue*.

5. *Combobox* yang digunakan untuk memilih *src_time_param* (kolom pencatat waktu perubahan pada tabel referensi) dan *dst_time_param* (kolom pencatat waktu perubahan pada tabel replika).
6. Tombol untuk menyimpan data *queue* baru.

3.2.3.4 Antarmuka Halaman Mengubah Pengaturan *Queue*

Antarmuka halaman mengubah pengaturan *queue* sama dengan antarmuka halaman tambah *queue*. Adapun yang membedakan adalah pada halaman mengubah pengaturan *queue* data sudah terisi berdasarkan pengaturan yang sudah disimpan sebelumnya.

3.2.3.5 Antarmuka Halaman Melihat Detail *Queue*

Antarmuka halaman melihat detail *queue* menampilkan data *queue* dan log jika terjadi kesalahan pada *queue*.

Gambar 3.15 merupakan rancangan antarmuka halaman melihat detail *queue*. Berikut penjelasan masing-masing nomor yang tertera pada Gambar 3.15.

1. Teks yang menampilkan data koneksi, schema, tabel
2. Tabel yang menampilkan pemetaan dari kolom pada tabel referensi ke kolom pada tabel replika.
3. Teks yang menampilkan frekuensi, *src_time_param* dan *dst_time_param*.
4. Tabel yang menampilkan log jika terjadi kesalahan pada *queue*.
5. Tombol untuk menuju halaman mengubah pengaturan *queue*.
6. Teks untuk menampilkan nama *queue*.
7. Tombol untuk mengunduh ZIP yang akan digunakan sebagai *listener*.

Queue :

EditZIP

Source

Connection

Schema

Table

Destination

Connection

Schema

Table

Source Column	Target Column

frequency

src_time_param

dsl_time_param

ID	Error	Created At

Gambar 3.15 Rancangan Antarmuka Halaman Melihat Detail Queue

BAB IV IMPLEMENTASI

Pada bab ini dijelaskan mengenai implementasi dari perancangan perangkat lunak. Implementasi yang dijelaskan meliputi lingkungan pembangunan perangkat lunak, implementasi antarmuka pengguna dan implementasi proses-proses yang terjadi pada masing-masing kasus penggunaan pada perangkat lunak. Implementasi sistem mengacu pada perancangan yang ditulis pada Bab 3. Namun, tidak menutup kemungkinan adanya perubahan-perubahan dari rancangan tersebut apabila memang diperlukan.

4.1 Lingkungan Implementasi

Dalam merancang perangkat lunak ini digunakan perangkat penduduk sebagai berikut.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem perangkat lunak berupa laptop dengan spesifikasi Asus. Intel Core™ i5-4200U CPU @ 1,60 GHz 2,30 GHz dan 8 GB *memory*.

4.1.2 Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk pengembangan sistem adalah sebagai berikut.

1. Microsoft Windows 10 sebagai sistem operasi.
2. Homestead versi 6.1.0 untuk virtualisasi vagrant dengan Ubuntu 16.04 sebagai sistem operasi.
3. JetBrains PhpStorm 2017.2.4 sebagai IDE untuk implementasi kerangka kerja.
4. XAMPP versi 3.2.2 sebagai *web server* tempat kerangka kerja dijalankan.
5. SQL Server 2016 sebagai basis data perangkat kerja.
6. Navicat Premium versi 11.2.6 sebagai *database client*

7. Google Chrome versi 62.0.3202.94 sebagai *web browser* antarmuka kerangka kerja Sync ITS.
8. PowerDesigner versi 16.5 untuk merancang basis data dan diagram perancangan perangkat lunak.
9. Draw.io untuk merancang rancangan antarmuka

Di dalam implementasinya nanti, Sync ITS membutuhkan beberapa variabel tetap sebagai pengaturan server. Variabel ini diletakkan di folder root Sync ITS dengan nama `.env`. Isi dari `.env` ditunjukkan pada Gambar 4.1.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

Gambar 4.1 Pengaturan Basis Data Pada File `.env`

Pada gambar terdapat variabel yang harus disesuaikan sebelum kerangka kerja dijalankan. Penjelasan masing-masing variabel sebagai berikut.

1. `DB_HOST`
Variabel alamat basis data Sync ITS.
2. `DB_PORT`
Variabel port untuk basis data Sync ITS.
3. `DB_DATABASE`
Variabel nama basis data yang dituju.
4. `DB_USERNAME`
Variabel *username* untuk *login*.
5. `DB_PASSWORD`
Variabel *password* untuk *login*.

4.2 Implementasi Basis Data

Pada sub bab ini akan dibahas mengenai implementasi dari rancangan basis data yang telah dibahas pada Bab 3.

4.2.1 Implementasi Tabel conns

Tabel conns merupakan tabel yang digunakan untuk menyimpan konfigurasi koneksi basis data. Implementasi tabel conns ditunjukkan pada Kode Sumber 4.1.

```
DROP TABLE [dbo].[conns]
GO
CREATE TABLE [dbo].[conns] (
[id] int NOT NULL IDENTITY(1,1) ,
[name] varchar(255) NULL ,
[property] text NULL ,
[created_at] datetime2(3) NULL DEFAULT (getdate()) ,
[updated_at] datetime2(3) NULL ,
[deleted_at] datetime2(3) NULL
)
```

Kode Sumber 4.1 Implementasi Tabel conns

4.2.2 Implementasi Tabel queue_lists

Tabel queue_lists merupakan tabel yang digunakan untuk menyimpan daftar tabel referensi dan tabel replikanya, nama *queue*, *mapping*, penjadwalan, dan waktu terakhir sinkronisasi. Implementasi tabel queue_lists seperti pada Kode Sumber 4.2.

```

CREATE TABLE [dbo].[queue_lists] (
[id] int NOT NULL IDENTITY(1,1) ,
[queue] varchar(255) NULL ,
[src_conn] int NULL ,
[src_table] varchar(255) NULL ,
[dst_conn] int NULL ,
[dst_table] varchar(255) NULL ,
[map] varchar(255) NULL ,
[last_sync] datetime2(3) NULL ,
[created_at] datetime2(3) NULL DEFAULT (getdate()) ,
[updated_at] datetime2(3) NULL ,
[deleted_at] datetime2(3) NULL ,
[frequency] varchar(25) NULL ,
[src_time_param] varchar(30) NULL ,
[dst_time_param] varchar(30) NULL ,
[src_schema] varchar(30) NULL ,
[dst_schema] varchar(30) NULL ,
[priority] int NULL ,
[self_reference] varchar(255) NULL DEFAULT NULL
)

```

Kode Sumber 4.2 Implementasi Tabel queue_lists

4.2.3 Implementasi Tabel queue_log

Tabel queue_log digunakan untuk mencatat *error* pada *queue* yang ditolak dan sudah melebihi maksimal perulangan oleh *listener*. Implementasi pembuatan tabel queue_log seperti pada Kode Sumber 4.3.

```

CREATE TABLE [dbo].[queue_log] (
[id] int NOT NULL IDENTITY(1,1) ,
[queue_list_id] int NULL ,
[queue_msg] text NULL ,
[created_at] datetime2(3) NULL DEFAULT (getdate()) ,
[updated_at] datetime2(3) NULL DEFAULT (getdate()) ,
[deleted_at] datetime2(3) NULL ,
[error_msg] text NULL
)

```

Kode Sumber 4.3 Implementasi Tabel queue_log

4.2.4 Implementasi Tabel users

Tabel users merupakan tabel yang digunakan untuk menyimpan data pengguna Sync ITS. Implementasi pembuatan tabel users seperti pada Kode Sumber 4.4.

```
CREATE TABLE [dbo].[users] (  
[id] int NOT NULL IDENTITY(1,1) ,  
[name] nvarchar(255) NOT NULL ,  
[email] nvarchar(255) NOT NULL ,  
[password] nvarchar(255) NOT NULL ,  
[remember_token] nvarchar(100) NULL ,  
[created_at] datetime NULL ,  
[updated_at] datetime NULL  
)
```

Kode Sumber 4.4 Implementasi Tabel users

4.3 Implementasi Proses

Pada sub bab ini akan dibahas tentang fungsi atau proses yang berjalan pada kerangka kerja Sync ITS.

4.3.1 Proses Menambah Koneksi

Proses ini merupakan proses yang dieksekusi ketika Administrator melakukan penambahan koneksi melalui halaman antarmuka. Implementasi proses menambah koneksi seperti pada Kode Sumber 4.5.

```
// App/Http/Controllers/ConnController.php

public function store(StoreConnectionRequest $request)
{
    $data = $request->except('_token');
    $data = $this->connService->store($data);
    if ($data ) {
        return redirect()->back()->with(["message" =>
"Connection has ben created"]);
    }
    return response()->json("Error", 500);
}
```

Kode Sumber 4.5 Implementasi Proses Menambah Koneksi

4.3.2 Proses Mengubah Koneksi

Proses ini merupakan proses yang dieksekusi ketika Administrator melakukan perubahan koneksi menggunakan halaman antarmuka. Kode Sumber 4.6 merupakan implementasi dari proses mengubah koneksi.

```
// App/Http/Controllers/ConnController.php

public function update(Request $request, $id)
{
    $input = $request->except('_token');

    $data = $this->connService->update($input, $id);

    return redirect()->back()->with(["message" =>
"Connection has been updated"]);
}
```

Kode Sumber 4.6 Implementasi Proses Mengubah Koneksi

4.3.3 Proses Menghapus Koneksi

Proses menghapus koneksi merupakan proses yang dieksekusi ketika Administrator menghapus koneksi melalui halaman antarmuka. Kode Sumber 4.7 merupakan implementasi dari proses menghapus koneksi.

```
// App/Http/Controllers/ConnController.php

public function destroy($id)
{
    if ($this->connService->destroy($id) ) {
        return
        redirect(route('connection.index'))->with(["message" =>
        "Connection has been deleted"]);
    } else {
        return redirect()->back()-
        >withErrors(["message" => "Something wrong happen"]);
    }
}
```

Kode Sumber 4.7 Implementasi Proses Menghapus Koneksi

4.3.4 Proses Menambah Queue

Proses ini merupakan proses yang dieksekusi ketika Administrator melakukan penambahan *queue* melalui halaman antarmuka. Kode Sumber 4.8 merupakan implementasi dari proses menambah *queue* baru.

```
// App/Http/Controller/Api/v1/QueueListController.php

public function store(CreateQueueRequest $request)
{
    $input = $request->except("_token");

    $data = $this->queueListService->store($input);

    if ($data) {
        return $this->response-
        >created(route("queueList.show", $data->id));
    }

    return $this->response->errorInternal();
}
```

Kode Sumber 4.8 Implementasi Proses Menambah Queue

4.3.5 Proses Mengubah Queue

Proses mengubah *queue* merupakan proses yang dieksekusi ketika Administrator menghapus *queue* melalui halaman antarmuka. Kode Sumber 4.9 merupakan implementasi dari proses mengubah *queue*.

```
public function update(Request $request, $id)
{
    $input = $request->except('_token');
    $data = $this->queueListService->update($input, $id);
    if ($data) {
        return $this->response-
        >accepted(route("queueList.show", $id));
    }
    return $this->response->errorInternal();
}
```

Kode Sumber 4.9 Implementasi Proses Mengubah Queue

4.3.6 Proses Menghapus Queue

Proses menghapus *queue* merupakan proses yang dieksekusi ketika Administrator menghapus *queue* melalui halaman antarmuka. Kode Sumber 4.10 merupakan implementasi dari proses yang menghapus *queue*.

```
public function destroy($id)
{
    $data = $this->queueListService->destroy($id);

    return $this->response->noContent();
}
```

Kode Sumber 4.10 Implementasi Proses Menghapus Queue

4.3.7 Proses Sinkronisasi Basis Data

Sync ITS menggunakan crontab untuk melakukan penjadwalan sinkronisasi. Crontab hanya bisa berjalan pada lingkungan Linux. Kode Sumber 4.12 merupakan implementasi penjadwalan dengan menggunakan crontab. Crontab akan menjalankan Kode Sumber 4.11 setiap 1 menit sekali.

```
php artisan schedule:run
```

Kode Sumber 4.11 Perintah Laravel untuk Penjadwalan

```
\* * * * * php /home/vagrant/Code/sync/artisan schedule:run
```

Kode Sumber 4.12 Crontab untuk Penjadwalan Sinkronisasi

```
protected function schedule(Schedule $schedule)
{
    $queueService = app()->make('queueListService');

    $queueLists = $queueService->getPriority();

    foreach ($queueLists as $queueList) {
        $schedule->job(new SyncJob($queueList->id))-
>cron($queueList->getOriginal('frequency'));
    }
}
```

Kode Sumber 4.13 Perintah Eksekusi Sinkronisasi

Kode Sumber 4.13 menjelaskan bahwa program akan meng-*query* data dan mengurutkannya berdasarkan prioritasnya. Program akan mengiterasi sekaligus memfilter *queue* mana yang harus dijalankan. *Scheduler* akan memfilter data berdasarkan nilai dari kolom frequency.

Setelah crontab menemukan ada *queue* yang harus diproses, maka pekerjaan SyncJob akan dijalankan. Kode Sumber 4.14 merupakan implementasi pada pekerjaan SyncJob.

```
public function handle()
{
    $this->syncService->sync($this->id);
}
```

Kode Sumber 4.14 Implementasi Pekerjaan SyncJob

Kode Sumber 4.14 akan menjalankan fungsi untuk sinkronisasi. Proses sinkronisasi dibagi menjadi beberapa bagian seperti berikut.

1. Sync ITS akan melakukan *pulling* ke tabel referensi berdasarkan waktu terakhir sinkronisasi. Catat waktu sebelum dilakukan *pulling*. Kode Sumber 4.15 merupakan sebagian dari implementasi tahap ini.

```
$now = TimeHelper::now();
$dataSource = $this->pull($queue,$columnMaster);
```

Kode Sumber 4.15 Sync ITS Melakukan *Pulling* ke Tabel Referensi

2. Jikalau ada data yang harus dikirimkan, maka data akan diproses. Jikalau tidak ada, proses sinkronisasi selesai.
3. Sync ITS mengecek apakah terdapat *self-reference*. Jikalau terdapat *self-reference*, maka data akan diurutkan dengan menggunakan algoritma *topological sorting*. Kode Sumber 4.16 merupakan sebagian dari implementasi tahap ini.

```
if ($queue->self_reference) {
    $sort = $this->topSort($queue, $dataSource);
    if ($sort) {
        $dataSource = collect($sort);
    }
}
```

Kode Sumber 4.16 Pengecekan *Self-Reference*

4. Data dibagi menjadi beberapa bagian (*chunking*), per bagian berisi 10 data. Maksimal pengulangan (*max_retry*) adalah jumlah bagian hasil *chunking*. Kode Sumber 4.17 merupakan implementasi dari tahap ini.

```
$chunks = $dataSource->chunk(10);
$maxRetry = count($chunks);
```

Kode Sumber 4.17 Proses *Chunking* Data

5. Sync ITS akan mempublish data ke RabbitMQ. Kode Sumber 4.18 merupakan sebagian dari implementasi tahap ini.

```
$this->rabbitService->publish(json_encode($payload),
    $queue->queue, $maxRetry);
```

Kode Sumber 4.18 Sync ITS Mempublish Data ke RabbitMQ

6. Perbarui kolom *last_sync* pada tabel *queue_lists* dengan nilai waktu pada nomor 1. Kode Sumber 4.19 merupakan sebagian implementasi dari tahap ini.

```
$this->queueListRepo->updateLastSync($id, $now);
```

Kode Sumber 4.19 Proses Perbarui Kolom *last_sync*

7. Data yang dipublish diterima oleh *listener*. *Listener* menerima data, lalu memproses data.
8. Jika data tidak ditemukan, maka tambah data baru. Jika data ditemukan tapi nilai hash-nya sama, maka lewati. Jika data ditemukan dan nilai hash-nya berbeda, maka perbarui data. Kode Sumber 4.20 merupakan implementasi dari penambahan data, Kode Sumber 4.21 merupakan implementasi dari pembaruan data.
9. Apabila terdeteksi *circular*, maka data ditambahkan ke tabel replika tanpa nilai *foreign key* nya. *Listener* akan mengirimkan pesan *ack* dan akan mempublish ulang

queue tersebut. Setiap *queue* mempunyai maksimal perulangan. Jika melebihi maksimal perulangan, maka *queue* akan ditolak dan tidak akan dipublish ulang. Kode Sumber 4.22, Kode Sumber 4.24 dan Kode Sumber 4.23 merupakan sebagian implementasi pada tahap ini.

10. Untuk kondisi *queue* ditolak dan tidak dipublish ulang (*dead letter queue*), *queue* akan dikirim ke *dead letter exchange* dengan nama *sync.dead* lalu *listener* akan mengirimkan perintah kepada Sync ITS untuk mencatat log yang berisi *queue_list_id*, isi *queue message* dan *error message*.

```
$db->insert($table, $map, $data);
```

Kode Sumber 4.20 Proses Tambah Data oleh Listener

```
$db->update($table, $map, $data, $pk);
```

Kode Sumber 4.21 Proses Perbarui Data oleh Listener

```
$msg->delivery_info["channel"]->basic_ack(  
$msg->delivery_info["delivery_tag"]);
```

Kode Sumber 4.22 Proses Pengiriman Acknowledgements

```
$channel->basic_nack(  
$msg->delivery_info['delivery_tag'], false, false);
```

Kode Sumber 4.23 Proses Pengiriman Negative Acknowledgements

```
$channel->basic_publish($msg, $exchange_name,  
$routingKey);
```

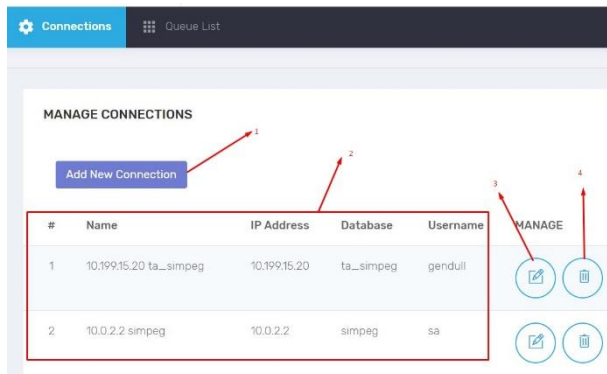
Kode Sumber 4.24 Proses Publish Ulang Queue

4.4 Implementasi Antarmuka

Pada sub bab ini akan dijelaskan implementasi antarmuka berdasarkan rancangan antarmuka yang telah dibahas pada Bab 3. Pada bagian ini juga akan dijelaskan mengenai fungsi-fungsi yang digunakan dalam program tugas akhir ini dan disertai dengan kode sumber masing-masing fungsi utama.

4.4.1 Antarmuka Halaman Menu Koneksi

Antarmuka halaman menu Koneksi ditunjukkan pada Gambar 4.2.



Gambar 4.2 Antarmuka Halaman Menu Koneksi

Berikut penjelasan masing-masing nomor yang tertera pada Gambar 4.2.

1. Tombol untuk menampilkan modal untuk membuat koneksi baru
2. Daftar koneksi yang sudah dibuat
3. Tombol untuk menampilkan modal untuk mengubah koneksi
4. Tombol untuk menghapus data

4.4.2 Antarmuka Halaman Tambah Koneksi

Antarmuka untuk menambah konfigurasi koneksi seperti pada Gambar 4.3

The screenshot shows a 'Create New Connection' dialog box with the following fields and annotations:

- 1**: Points to the 'Connection Name' text input field.
- 2**: Points to the 'Driver' dropdown menu, which currently shows 'sqlsrv'.
- 3**: Points to the 'Host' text input field.
- 4**: Points to the 'Port' text input field, which contains the value '1433'.
- 5**: Points to the 'Database' text input field.
- 6**: Points to the 'Username' text input field.
- 7**: Points to the 'Password' text input field.
- 8**: Points to the 'Close' button at the bottom right.
- 9**: Points to the 'Create' button at the bottom left.

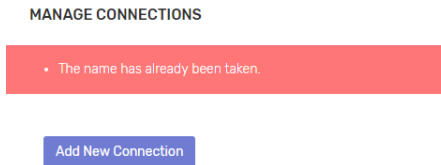
Additional text in the dialog includes 'Connection name must be unique' below the name field and a small 'x' icon in the top right corner.

Gambar 4.3 Antarmuka untuk Menambah Koneksi

Berikut penjelasan masing-masing nomor yang tertera pada Gambar 4.3.

1. Merupakan tempat memasukkan nama koneksi baru. Nama koneksi tidak boleh sama dengan yang sudah ada.
2. Merupakan tempat untuk memilih *driver* yang dipakai.
3. Merupakan tempat untuk memasukkan alamat koneksi.
4. Merupakan tempat untuk memasukkan *port* koneksi
5. Merupakan tempat untuk memasukkan nama basis data
6. Merupakan tempat untuk memasukkan username
7. Merupakan tempat untuk memasukkan *password*

8. Merupakan tombol untuk membatalkan pembuatan koneksi baru
 9. Merupakan tombol untuk menyimpan data koneksi baru
- Jika Administrator memasukkan nama koneksi yang sama pada, hasilnya adalah seperti Gambar 4.4.



Gambar 4.4 Tampilan Info jika Nama Koneksi Sama

4.4.3 Antarmuka Halaman Menu Daftar Queue

Gambar 4.5 merupakan antarmuka halaman yang menampilkan seluruh *queue*.

MANAGE QUEUE

1 Add New Queue 2 Update Priority

#	Queue Name	Source Connection	Source Table	Dest Connection	Dest Table	Frequency (Minute)	Last Sync	MANAGE
1	queue_1	10.0.2.2 simpang	glaksheder	10.0.2.2 okantor	glaksheder	1	2018-01-08 08:38:26.000	4 5
2	queue_2	10.0.2.2 simpang	unit	10.0.2.2 okantor	unit	1	2018-01-08 08:38:25.000	4 5

Gambar 4.5 Antarmuka Halaman Daftar Queue

Berikut penjelasan masing-masing nomor yang tertera pada Gambar 4.5.

1. Tombol untuk menuju halaman tambah *queue*
2. Tombol untuk memperbarui prioritas *queue*
3. Daftar *queue* yang sudah pernah dibuat
4. Tombol untuk menuju halaman mengubah pengaturan *queue*
5. Tombol untuk menghapus *queue*.

4.4.4 Antarmuka Halaman Tambah Queue

Antarmuka halaman tambah *queue* merupakan tampilan apabila Administrator ingin menambah *queue*. Gambar 4.6 merupakan tampilan antarmuka halaman tambah *queue*.

The screenshot displays a web form for adding a queue, divided into two main sections: SOURCE and DESTINATION. The form includes dropdown menus for Connection, Schema, and Table in both sections. Below these, there are input fields for Source Column and Target Column, with a message 'No matching records found' below them. Further down, there is a field for Frequency (minutes) set to 5, and two more dropdown menus for Source Time Parameter and Destination Time Parameter. A submit button is located at the bottom left. Numbered arrows point to specific elements: 1 points to the SOURCE Connection dropdown, 2 points to the SOURCE Schema dropdown, 3 points to the SOURCE Table dropdown, 4 points to the Source Column input field, 5 points to the Target Column input field, 6 points to the Frequency (minutes) input field, and 7 points to the Destination Time Parameter dropdown.

Gambar 4.6 Antarmuka Halaman Tambah *Queue*

Berikut penjelasan masing-masing nomor pada Gambar 4.6.

1. *Combobox* untuk memilih koneksi.
2. *Combobox* untuk memilih skema.
3. *Combobox* untuk memilih tabel.

4. Daftar kolom pada tabel referensi dan tabel replika.
5. *Combobox* untuk memilih kolom pada tabel replika yang sesuai dengan kolom referensi.
6. *Textbox* untuk memasukkan frekuensi penjadwalan.
7. *Combobox* untuk memilih kolom parameter waktu.

4.4.5 Antarmuka Halaman Mengubah Queue

Antarmuka halaman mengubah pengaturan *queue* sama dengan antarmuka halaman tambah *queue*. Adapun yang membedakan adalah pada halaman mengubah pengaturan *queue* data sudah terisi berdasarkan pengaturan yang sudah disimpan sebelumnya.

4.4.6 Antarmuka Halaman Melihat Detail Queue

Antarmuka halaman melihat detail *queue* merupakan tampilan apabila Administrator melihat detail *queue*. Gambar 4.7 merupakan implementasi halaman melihat *queue*.

Penjelasan masing-masing nomor pada Gambar 4.7 sebagai berikut.

1. Nama *queue*.
2. *Combobox* aksi yang berisi unduh ZIP *listener* dan ubah data
3. Data pengaturan *queue*.
4. Tampilan *error log*

Queue : 1512460315_1_stakeholder_t2_stakeholder

Action ▾

1

2

SOURCE

Connection

10.0.2.2 simpeg

Schema

dbo

Table

stakeholder

DESTINATION

Connection

10.0.2.2 simpeg

Schema

dbo

Table

stakeholder

Source Column

Target Column

id

id

name

name

code

code

enabled

enabled

parent_id

parent_id

created_at

created_at

updated_at

updated_at

level

level

unit_id

unit_id

Frequency (minutes)

1

Source Time Parameter

updated_at

Show 10 ▾ entries

4

3

Id	Error	Created at ▾
42	[\"SQLSTATE[23000]: [Microsoft][ODBC Driver 13 for SQL Server][SQL Server]The INSERT statement conflicted with the FOREIGN KEY constraint 'FK____stakehold____unit____2E1BDC42'. The conflict occurred in database 'Yekantor\\', table 'Ydbo.unit\\', column 'id'.\"]	2018-01-08 03:36:04.000
41	[\"SQLSTATE[23000]: [Microsoft][ODBC Driver 13 for SQL Server][SQL Server]The INSERT statement conflicted with the FOREIGN KEY constraint 'FK____stakehold____unit____2E1BDC42'. The conflict occurred in database 'Yekantor\\', table 'Ydbo.unit\\', column 'id'.\"]	2018-01-08 03:36:02.000
37	[\"SQLSTATE[42S22]: [Microsoft][ODBC Driver 13 for SQL Server][SQL Server]Invalid column name 'bodee'.\"]	2017-12-11 16:24:42.000
27	\"foreign key same table\"	2017-12-11 12:05:01.000
24	\"error\"	2017-12-11 11:41:06.000
23	\"error\"	2017-12-11 11:41:06.000

Showing 1 to 6 of 6 entries

Previous 1 Next

Gambar 4.7 Antarmuka Halaman Melihat Queue

BAB V

HASIL UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan uji coba yang dilakukan pada aplikasi yang telah dikerjakan serta analisis dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, skenario uji coba yang meliputi uji kebenaran dan uji kinerja serta analisis setiap pengujian.

5.1 Lingkungan Uji Coba

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji pengiriman data berdasarkan pengaturan yang dibuat pada tugas akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dibedakan menjadi 3 jenis yaitu *source* seperti pada Tabel 5.1, *target* seperti pada Tabel 5.3 dan server Sync ITS seperti pada Tabel 5.2.

Tabel 5.1 Tabel Spesifikasi Lingkungan Pengujian *Source*

Spesifikasi	Deskripsi
Prosesor	Intel Core™ i5-4200U CPU @ 1,60 GHz 2,30 GHz
Memory (RAM)	8 GB
Tipe sistem	64-bit
Sistem operasi	Windows 10 Pro
Basis data	SQL Server 2016

Tabel 5.2 Tabel Spesifikasi Lingkungan Pengujian Server Sync ITS

Spesifikasi	Deskripsi
Prosesor	Intel Core™ i5-4200U CPU @ 1,60 GHz 2,30 GHz
Memory (RAM)	8 GB
Tipe sistem	64-bit
Sistem operasi	Windows 10 Pro
Basis data	SQL Server 2016

Tabel 5.3 Tabel Spesifikasi Lingkungan Pengujian Target

Spesifikasi	Deskripsi
Prosesor	Intel Core™ i3-2100 CPU @ 3,10 GHz 3,10 GHz
Memory (RAM)	8 GB
Tipe sistem	64-bit
Sistem operasi	Windows 10 Pro
Basis data	SQL Server 2016

5.2 Dasar Pengujian

Pengujian yang dilakukan pada tugas akhir ini berupa pengujian fungsionalitas. Pengujian fungsionalitas dilakukan dengan model *black box* untuk menguji pengelolaan koneksi, *queue*, dan proses sinkronisasi basis data. Pengujian ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi pada tahap kebutuhan benar-benar diimplementasikan dan bekerja seperti yang diharapkan.

5.3 Pengujian Fungsionalitas

Sub bab ini menjelaskan tentang skenario pengujian fungsionalitas perangkat lunak pada tugas akhir ini. Pengujian didokumentasikan secara sistematis sebagai tolok ukur keberhasilan sistem.

5.3.1 Pengujian Pengelolaan Koneksi Basis Data

Pengujian pengelolaan koneksi basis data dibagi menjadi beberapa proses yaitu membuat, mengubah, dan menghapus pengaturan. Skenario Pengujian menambah koneksi dijelaskan pada Tabel 5.4, pengujian mengubah koneksi pada Tabel 5.5, sedangkan pengujian menghapus koneksi pada Tabel 5.6.

Tabel 5.4 Skenario Pengujian Membuat Koneksi Baru

Nomor	PE-001
Nama	Membuat koneksi baru
Tujuan	Memeriksa fungsi membuat koneksi berfungsi atau tidak
Kondisi awal	Tidak ada koneksi yang disimpan
Skenario	<ol style="list-style-type: none"> 1. Administrator memilih tombol tambah koneksi baru 2. Administrator mengisi nama koneksi, driver, alamat basis data, <i>port</i>, <i>username</i>, <i>password</i>. 3. Administrator menyimpan pengaturan dengan menekan tombol simpan.
Masukan	Koneksi baru
Keluaran yang diharapkan	Konfigurasi koneksi tersimpan pada tabel <i>conns</i> .
Hasil Pengujian	Berhasil

5.3.2 Pengujian Pengelolaan Pengaturan Queue

Pengujian pengelolaan *queue* pada basis data dibagi menjadi beberapa proses yaitu membuat, mengubah, dan menghapus pengaturan. Skenario Pengujian menambah *queue* dijelaskan pada Tabel 5.7, pengujian mengubah *queue* pada Tabel 5.8, sedangkan menghapus *queue* pada Tabel 5.9.

Tabel 5.5 Skenario Pengujian Mengubah Konfigurasi Koneksi

Nomor	PE-002
Nama	Mengubah konfigurasi koneksi
Tujuan	Memeriksa fungsi mengubah koneksi berfungsi atau tidak
Kondisi awal	Koneksi sebelumnya sudah tersimpan
Skenario	<ol style="list-style-type: none"> 1.Administrator memilih tombol edit berlambang pensil pada daftar koneksi. 2.Administrator mengisi nama koneksi, driver, alamat basis data, <i>port</i>, <i>username</i>, <i>password</i> yang berbeda dengan data sebelumnya. 3.Administrator menyimpan pengaturan dengan menekan tombol simpan.
Masukan	Perubahan konfigurasi koneksi.
Keluaran yang diharapkan	Konfigurasi koneksi baru berhasil diperbarui dan tersimpan pada tabel conns.
Hasil Pengujian	Berhasil

Tabel 5.6 Skenario Pengujian Menghapus Koneksi

Nomor	PE-003
Nama	Menghapus koneksi
Tujuan	Memeriksa fungsi menghapus koneksi berfungsi atau tidak
Kondisi awal	Koneksi sebelumnya sudah tersimpan
Skenario	<ol style="list-style-type: none"> 1. Administrator memilih tombol hapus berlambang tempat sampah pada daftar koneksi. 2. Administrator memilih jawaban “ya” saat dikonfirmasi untuk menghapus koneksi tersebut.
Masukan	Penghapusan koneksi
Keluaran yang diharapkan	Koneksi berhasil dihapus
Hasil Pengujian	Berhasil

Tabel 5.7 Skenario Pengujian Penambahan *Queue*

Nomor	PE-004
Nama	Membuat <i>queue</i> baru
Tujuan	Memeriksa fungsi membuat <i>queue</i> berfungsi atau tidak
Kondisi awal	Tidak ada <i>queue</i> yang disimpan
Skenario	<ol style="list-style-type: none"> 1. Administrator memilih tombol tambah <i>queue</i> baru 2. Administrator mengisi form 3. Administrator menyimpan pengaturan <i>queue</i> dengan menekan tombol simpan.
Masukan	<i>Queue</i> baru
Keluaran yang diharapkan	Konfigurasi <i>queue</i> tersimpan pada tabel <i>queue_list</i> .
Hasil Pengujian	Berhasil

Tabel 5.8 Skenario Pengujian Mengubah *Queue*

Nomor	PE-005
Nama	Mengubah <i>queue</i> koneksi
Tujuan	Memeriksa fungsi mengubah <i>queue</i> berfungsi atau tidak
Kondisi awal	<i>Queue</i> sebelumnya sudah tersimpan
Skenario	<ol style="list-style-type: none"> 1. Administrator memilih tombol edit berlambang pensil pada daftar <i>queue</i>. 2. Administrator mengisi <i>form</i> dengan data yang berbeda dengan data sebelumnya. 3. Administrator menyimpan pengaturan dengan menekan tombol simpan.
Masukan	Perubahan konfigurasi <i>queue</i> .
Keluaran yang diharapkan	Konfigurasi <i>queue</i> baru berhasil diperbarui dan tersimpan pada tabel <i>queue_lists</i> .
Hasil Pengujian	Berhasil

Tabel 5.9 Skenario Pengujian Menghapus *Queue*

Nomor	PE-006
Nama	Menghapus <i>queue</i>
Tujuan	Memeriksa fungsi menghapus <i>queue</i> berfungsi atau tidak
Kondisi awal	<i>Queue</i> sebelumnya sudah tersimpan
Skenario	<ol style="list-style-type: none"> 1. Administrator memilih tombol hapus berlambang tempat sampah pada daftar <i>queue</i>. 2. Administrator memilih jawaban “ya” saat dikonfirmasi untuk menghapus <i>queue</i> tersebut.
Masukan	Penghapusan <i>queue</i>
Keluaran yang diharapkan	<i>Queue</i> berhasil dihapus
Hasil Pengujian	Berhasil

5.3.3 Pengujian Pembaruan Prioritas *Queue*

Nomor	PE-007
Nama	Memperbarui prioritas <i>queue</i>
Tujuan	Memeriksa fungsi memperbarui prioritas <i>queue</i> berfungsi atau tidak
Kondisi awal	Prioritas <i>queue</i> masih kosong
Skenario	<ol style="list-style-type: none"> 1. Administrator memilih tombol <i>update priority</i>
Masukan	Penghapusan <i>queue</i>
Keluaran yang diharapkan	<i>Queue</i> berhasil dihapus
Hasil Pengujian	Berhasil

id	queue	priority
1	1512460321_1_stakeholder_12_stakeholder	(Null)
36	1512460315_1_stakeholder_12_stakeholder	(Null)
38	1512464521_3_unit_12_unit	(Null)
47	1512527533_3_users_12_usersx	(Null)
1045	1512616426_1013_unit_1012_unit	(Null)

(a)

id	queue	priority
1	1512460321_1_stakeholder_12_stakeholder	0
36	1512460315_1_stakeholder_12_stakeholder	1
38	1512464521_3_unit_12_unit	0
47	1512527533_3_users_12_usersx	0
1045	1512616426_1013_unit_1012_unit	0





(b)

Gambar 5.1 (a, b) Ilustrasi Pembaruan Prioritas *Queue*

5.3.4 Pengujian Sinkronisasi Basis Data

Uji coba ini dilakukan untuk menguji apakah fungsionalitas pengiriman data telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya, serta akan dihitung selisih antara total waktu aplikasi berjalan dan waktu untuk memproses data (*overhead*). Uji coba akan didasarkan pada beberapa skenario untuk menguji validasi pengiriman data Sync ITS. Skenario pengujian terdiri dari 5 bagian yaitu sebagai berikut.

1. Skenario pengiriman data awal.
2. Skenario pengiriman perubahan data.
3. Skenario sinkronisasi data pada tabel yang memiliki relasi terhadap tabel lain (prioritas).
4. Skenario sinkronisasi data pada tabel yang memiliki relasi terhadap tabel yang sama (*self-reference*).
5. Skenario melanjutkan sinkronisasi yang terputus.
6. Skenario kegagalan sinkronisasi

#	Queue Name	Source Connection	Source Table	Dest Connection	Dest Table	Frequency (Minute)	Last Sync	MANAGE
1	1612440316_1_stakeholder_12_stakeholder	10.107.255.221 simpeg	stakeholder	10.107.255.222 ekantor	stakeholder	1		 
2	1612444521_3_unit_12_unit	10.107.255.221 simpeg	unit	10.107.255.222 ekantor	unit	1		 

Gambar 5.2 Pengaturan *Queue* untuk Pengujian

5.3.4.1 Skenario Uji Coba 1

Skenario uji coba 1 merupakan skenario pengiriman data awal dari salah satu tabel. Kondisi ini merupakan sinkronisasi pertama antar tabel referensi ke tabel replika. Tabel yang digunakan adalah tabel unit dengan jumlah data 184 baris. Jumlah baris yang akan dikirimkan per *chunk* adalah 10 baris, sehingga total ada 19 *chunk*. *Listener* mendeteksi urutan *chunk* dengan menggunakan *delivery_tag* pada *queue header*. Skenario pengujian dijelaskan pada Tabel 5.10. Ilustrasi sinkronisasi diperlihatkan pada Gambar 5.3, Gambar 5.4, dan Gambar 5.5. Tabel 5.11 merupakan data berapa kali *queue* diproses saat sinkronisasi.

Tabel 5.10 Skenario Uji Coba 1

Nomor	PE-008
Nama	Sinkronisasi data awal
Tujuan	Memeriksa fungsi pengiriman data awal dari tabel referensi ke tabel replika berfungsi atau tidak
Kondisi awal	Data tabel referensi masih kosong
Skenario	Administrator menjalankan perintah sinkronisasi
Masukan	Perintah sinkronisasi
Keluaran yang diharapkan	Isi tabel referensi sama dengan tabel replika
Hasil Pengujian	Berhasil

name	level	code	enabled	updated_at	created_at	id
Tanpa Unit	1	-	1	2017-12-15 07:28:42.0000000	2017-12-15 05:28:42.0433333	0
Rektorat	1	IT2	1	2017-12-15 07:28:42.0000000	2017-12-15 05:28:42.1166667	1
Senat Akademik	2		1	2017-12-15 05:28:42.1666667	2017-12-15 05:28:42.1666667	2
Majelis Wali Amanat	2	IT2.VI	1	2017-12-15 05:28:42.2000000	2017-12-15 05:28:42.2000000	3
Sekretaris Institut	2	IT2.VII	1	2017-12-15 05:28:42.2500000	2017-12-15 05:28:42.2500000	4
Satuan Pengawas Internal	2	IT2.VIII	1	2017-12-15 05:28:42.2700000	2017-12-15 05:28:42.2700000	5
Fakultas Matematika dan Ilmu Per 3	3	IT2.1	1	2017-12-15 05:28:42.2900000	2017-12-15 05:28:42.2900000	6
Fakultas Teknologi Industri (FTI)	3		1	2017-12-15 05:28:42.3233333	2017-12-15 05:28:42.3233333	7
Fakultas Teknik Sipil dan Perencar 3	3	IT2.3	1	2017-12-15 05:28:42.3666667	2017-12-15 05:28:42.3666667	8
Fakultas Teknologi Kelautan (FTK)	3		1	2017-12-15 05:28:42.4000000	2017-12-15 05:28:42.4000000	9

SELECT TOP 1000 * FROM [dbo].[unit] Record 2 of 184 in page 1

Gambar 5.3 Isi Tabel Unit (Referensi)

name	level	code	enabled	updated_at	created_at	id
(Null)	(Null)	(Null)	(Null)	(Null)	(Null)	(Null)

Gambar 5.4 Isi Tabel Unit (Replika) Sebelum Sinkronisasi

name	level	code	enabled	updated_at	created_at	id
Tanpa Unit	1	-	1	2017-12-15 07:28:42.0000000	2017-12-15 05:28:42.0433333	0
Rektorat	1	IT2	1	2017-12-15 07:28:42.0000000	2017-12-15 05:28:42.1166667	1
Senat Akademik	2		1	2017-12-15 05:28:42.1666667	2017-12-15 05:28:42.1666667	2
Majelis Wali Amanat	2	IT2.VI	1	2017-12-15 05:28:42.2000000	2017-12-15 05:28:42.2000000	3
Sekretaris Institut	2	IT2.VII	1	2017-12-15 05:28:42.2500000	2017-12-15 05:28:42.2500000	4
Satuan Pengawas Internal	2	IT2.VIII	1	2017-12-15 05:28:42.2700000	2017-12-15 05:28:42.2700000	5
Fakultas Matematika dan Ilmu Per 3	3	IT2.1	1	2017-12-15 05:28:42.2900000	2017-12-15 05:28:42.2900000	6
Fakultas Teknologi Industri (FTI)	3		1	2017-12-15 05:28:42.3233333	2017-12-15 05:28:42.3233333	7
Fakultas Teknik Sipil dan Perencar 3	3	IT2.3	1	2017-12-15 05:28:42.3666667	2017-12-15 05:28:42.3666667	8
Fakultas Teknologi Kelautan (FTK)	3		1	2017-12-15 05:28:42.4000000	2017-12-15 05:28:42.4000000	9

SELECT TOP 1000 * FROM [dbo].[unit] Record 8 of 184 in page 1

Gambar 5.5 Isi Tabel Unit (Replika) setelah Sinkronisasi

5.3.4.2 Skenario Uji Coba 2

Skenario uji coba 2 merupakan skenario pengiriman perubahan data. Tabel yang digunakan adalah tabel unit dengan jumlah perubahan 2 baris. Jumlah baris yang akan dikirimkan per *chunk* adalah 10 baris, sehingga total ada 1 *chunk*. *Listener* mendeteksi urutan *chunk* dengan menggunakan *delivery_tag* pada *queue header*. Skenario pengujian dijelaskan pada Tabel 5.12 Tabel 5.10. Ilustrasi sinkronisasi diperlihatkan pada Gambar 5.6, Gambar 5.7, dan Gambar 5.8. Tabel 5.13 merupakan log berapa kali *queue* diproses saat sinkronisasi.

Tabel 5.11 Jumlah Pemrosesan *Queue* Uji Coba 1

<i>Delivery Tag</i>	Jumlah	Waktu Proses (detik)
1	1	0,068997622
2	1	0,057002783
3	1	0,083241224
4	1	0,056241035
5	1	0,059452057
6	1	0,050045729
7	1	0,057916164
8	1	0,096057177
9	1	0,060996056
10	1	0,061323881
11	1	0,065600157
12	1	0,050021887
13	1	0,079592943
14	1	0,078892946
15	1	0,086414814
16	1	0,066524267
17	1	0,05420661
18	1	0,054302931
19	1	0,018675089
Total Per Data		1,205505371
Total Waktu Pemrosesan		1,233798742
<i>Overhead</i>		0.028293

Tabel 5.12 Skenario Uji Coba 2

Nomor	PE-009
Nama	Sinkronisasi perubahan data.
Tujuan	Memeriksa fungsi pengiriman perubahan data dari tabel referensi ke tabel replika berfungsi atau tidak.
Kondisi awal	Data tabel referensi sudah terisi.
Skenario	Administrator menjalankan perintah sinkronisasi
Masukan	Perintah sinkronisasi.
Keluaran yang diharapkan	Isi tabel referensi sama dengan tabel replika.
Hasil Pengujian	Berhasil.

name	level	code	enabled	updated_at	created_at	id
Tanpa Unit Contoh Edit	1	-	1	2017-12-15 07:28:42.0000000	2017-12-15 05:28:42.0433333	0
Rektorat Contoh Edit	1	IT2	1	2017-12-15 07:28:42.0000000	2017-12-15 05:28:42.1166667	1
Senat Akademik	2		1	2017-12-15 05:28:42.1666667	2017-12-15 05:28:42.1666667	2
Majelis Wali Amanat	2	IT2.VI	1	2017-12-15 05:28:42.2000000	2017-12-15 05:28:42.2000000	3
Sekretaris Institut	2	IT2.VII	1	2017-12-15 05:28:42.2500000	2017-12-15 05:28:42.2500000	4
Satuan Pengawas Internal	2	IT2.VIII	1	2017-12-15 05:28:42.2700000	2017-12-15 05:28:42.2700000	5
Fakultas Matematika dan Ilmu Per 3		IT2.1	1	2017-12-15 05:28:42.2900000	2017-12-15 05:28:42.2900000	6
Fakultas Teknologi Industri (FTI)	3		1	2017-12-15 05:28:42.3233333	2017-12-15 05:28:42.3233333	7
Fakultas Teknik Sipil dan Perencer		IT2.3	1	2017-12-15 05:28:42.3666667	2017-12-15 05:28:42.3666667	8
Fakultas Teknologi Kelautan (FTK)	3		1	2017-12-15 05:28:42.4000000	2017-12-15 05:28:42.4000000	9

Gambar 5.6 Data pada Tabel Referensi

name	level	code	enabled	updated_at	created_at	id
Tanpa Unit Contoh Edit	1	-	1	2017-12-15 07:28	2017-12-15 05:2	0
Rektorat Contoh Edit	1	IT2	1	2017-12-15 07:28	2017-12-15 05:2	1

Gambar 5.7 Perubahan Data

name	level	code	enabled	updated_at	created_at	id
Tanpa Unit Contoh Edit	1	-	1	2017-12-15 07:28:42.0000000	2017-12-15 05:28:42.0433333	0
Rektorat Contoh Edit	1	IT2	1	2017-12-15 07:28:42.0000000	2017-12-15 05:28:42.1166667	1
Senat Akademik	2		1	2017-12-15 05:28:42.1666667	2017-12-15 05:28:42.1666667	2
Majelis Wali Amanat	2	IT2.VI	1	2017-12-15 05:28:42.2000000	2017-12-15 05:28:42.2000000	3
Sekretaris Institut	2	IT2.VII	1	2017-12-15 05:28:42.2500000	2017-12-15 05:28:42.2500000	4
Satuan Pengawas Internal	2	IT2.VIII	1	2017-12-15 05:28:42.2700000	2017-12-15 05:28:42.2700000	5
Fakultas Matematika dan Ilmu Per 3		IT2.1	1	2017-12-15 05:28:42.2900000	2017-12-15 05:28:42.2900000	6

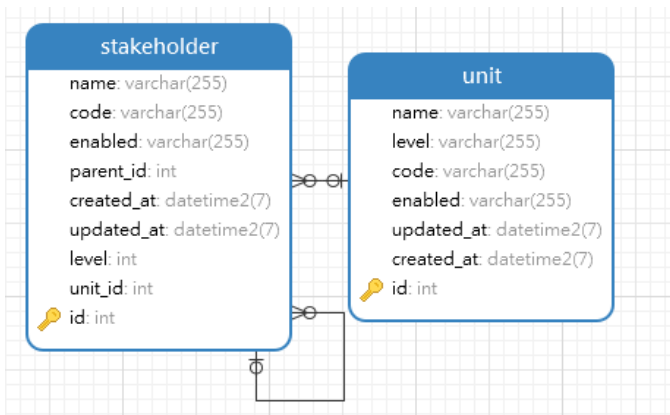
Gambar 5.8 Hasil pada Tabel Replika

Tabel 5.13 Jumlah Pemrosesan per Queue Uji Coba 2

Delivery Tag	Jumlah diproses	Waktu Proses (detik)
1	1	0,020329704
Total Per Data		0,020329704
Total Waktu Pemrosesan		0,030311210
<i>Overhead</i>		0,009981506

5.3.4.3 Skenario Uji Coba 3

Skenario uji coba 3 adalah sinkronisasi data pada tabel yang memiliki relasi terhadap tabel lain (prioritas). Tabel yang digunakan pada skenario ini adalah tabel unit dan stakeholder. Relasi stakeholder dan unit seperti pada Gambar 5.9. Tabel stakeholder berisi 100 data (10 *chunk*), sedangkan tabel unit berisi 184 data (19 *chunk*). Skenario dijelaskan pada Tabel 5.14. Gambar 5.11 dan Gambar 5.3 merupakan data pada tabel referensi. Gambar 5.4 dan Gambar 5.10 merupakan data pada tabel replika sebelum sinkronisasi. Gambar 5.5 dan Gambar 5.12 merupakan tabel replika setelah sinkronisasi. Tabel 5.15 merupakan tabel yang menunjukkan berapa kali *queue* diproses oleh *listener*.

**Gambar 5.9 Relasi Stakeholder dan Unit**

Tabel 5.14 Skenario Uji Coba 3

Nomor	PE-010
Nama	Sinkronisasi tabel yang berelasi.
Tujuan	Memeriksa fungsi pengiriman data berdasarkan prioritas berfungsi atau tidak.
Kondisi awal	Data tabel referensi sudah terisi. Tabel replika masih kosong.
Skenario	Administrator menjalankan perintah sinkronisasi
Masukan	Perintah sinkronisasi.
Keluaran yang diharapkan	Isi tabel referensi sama dengan tabel replika.
Hasil Pengujian	Berhasil.

name	code	enabled	parent_id	created_at	updated_at	level	unit_id	id
(Null)	(Null)	(Null)	(Null)	(Null)	(Null)	(Null)	(Null)	(Null)

Gambar 5.10 Data Tabel Stakeholder (Replika) sebelum Sinkronisasi

id	name	code	enabled	parent_id	created_at	updated_at	level	unit_id	id
1	Rektor	IT2	0	(Null)	2016-11-22 11:47:15.000000	2016-11-29 16:31:12.000000	(Null)	1	1
2	Wakil Rektor I	IT2.I	0	1	2016-11-22 11:47:16.000000	2016-11-22 11:47:16.000000	(Null)	1	2
3	Wakil Rektor II	IT2.II	0	1	2016-11-22 11:47:15.000000	2016-11-22 11:47:15.000000	(Null)	1	3
4	Wakil Rektor III	IT2.III	0	15	2016-11-22 11:47:16.000000	2016-11-22 11:47:16.000000	(Null)	1	4
5	Wakil Rektor IV	IT2.IV	0	1	2016-11-22 11:47:16.000000	2016-11-22 11:47:16.000000	(Null)	1	5
6	Ketua Senat Akademik	IT2.V	0	1	2016-11-22 11:47:15.000000	2016-11-22 11:47:15.000000	(Null)	2	6
9	Ketua Majelis Wali Amanat	IT2.VI	0	(Null)	2016-11-22 11:47:15.000000	2017-05-17 11:12:25.000000	(Null)	3	9
11	Ketua Dewan Pengawas	IT2.VII	0	1	2016-11-22 11:47:16.000000	2016-11-22 11:47:16.000000	(Null)	4	11

Gambar 5.11 Data Tabel Stakeholder (Referensi)

name	code	enabled	parent_id	created_at	updated_at	level	unit_id	id
Rektor	IT2	0	(Null)	2016-11-22 11:47:15.000000	2016-11-29 16:31:12.000000	(Null)	1	1
Wakil Rektor I	IT2.I	0	1	2016-11-22 11:47:16.000000	2016-11-22 11:47:16.000000	(Null)	1	2
Wakil Rektor II	IT2.II	0	1	2016-11-22 11:47:15.000000	2016-11-22 11:47:15.000000	(Null)	1	3
Wakil Rektor III	IT2.III	0	15	2016-11-22 11:47:16.000000	2016-11-22 11:47:16.000000	(Null)	1	4
Wakil Rektor IV	IT2.IV	0	1	2016-11-22 11:47:16.000000	2016-11-22 11:47:16.000000	(Null)	1	5
Ketua Senat Akademik	IT2.V	0	1	2016-11-22 11:47:15.000000	2016-11-22 11:47:15.000000	(Null)	2	6
Ketua Majelis Wali Amanat	IT2.VI	0	(Null)	2016-11-22 11:47:15.000000	2017-05-17 11:12:25.000000	(Null)	3	9
Ketua Dewan Pengawas	IT2.VII	0	1	2016-11-22 11:47:16.000000	2016-11-22 11:47:16.000000	(Null)	4	11

Gambar 5.12 Data Tabel Stakeholder (Replika) Setelah Sinkronisasi

Tabel 5.15 Jumlah Pemrosesan Stakeholder per *Queue* Uji Coba 3

Delivery Tag	Jumlah Proses	Waktu Proses (detik)
1	1	0,127249969
2	1	0,034129903
3	1	0,045481235
4	1	0,034972972
5	1	0,031919003
6	1	0,046865566
7	1	0,031229704
8	1	0,017913134
9	1	0,031101568
10	1	0,028483561
Total Per Data1		0,42934662
Total Waktu Pemrosesan		0,43211234
<i>Overhead</i>		0,002766

5.3.4.4 Skenario Uji Coba 4

Skenario uji coba 4 adalah sinkronisasi data pada tabel yang memiliki relasi terhadap diri sendiri (*self-reference*). Tabel 5.16 merupakan skenario pengujian uji coba 4. Tabel yang digunakan pada uji coba ini adalah tabel stakeholder. Kolom *parent_id* merupakan kolom yang mereferensi kolom *id*. PDM tabel stakeholder seperti Data tabel stakeholder dimodifikasi agar mempunyai data *circular*, yaitu data dengan *id* 4 dan 15. Gambar 5.9. Ilustrasi skenario uji coba 4 seperti Gambar 5.13, Gambar 5.10 dan Gambar 5.14. Tabel 5.17 merupakan tabel pemrosesan data per *queue*. Tabel 5.17 menunjukkan bahwa *delivery tak* 1 diproses 2 kali, karena ketika proses penambahan data, data dengan *id* 15 belum masuk. Sehingga dilakukan penambahan data tanpa *foreign key* lalu *queue* dipublish ulang dan diletakkan pada urutan *queue* terakhir. Pada iterasi kedua, data dengan *id* 4 diperbarui. Data dengan *id* 15 hanya diproses sekali karena data dengan *id* 4 sudah masuk pada iterasi pertama.

Tabel 5.16 Skenario Uji Coba 4

Nomor	PE-011
Nama	Sinkronisasi tabel yang berelasi terhadap diri sendiri.
Tujuan	Memeriksa fungsi pengiriman data yang berelasi terhadap diri sendiri berfungsi atau tidak.
Kondisi awal	Data tabel referensi sudah terisi. Tabel replika masih kosong.
Skenario	Administrator menjalankan perintah sinkronisasi
Masukan	Perintah sinkronisasi.
Keluaran yang diharapkan	Isi tabel referensi sama dengan tabel replika.
Hasil Pengujian	Berhasil.

id	name	code	enabled	parent_id	created_at	updated_at	level	unit_id
1	Rektor	IT2	0	(Null)	2016-11-22 11:47:15.0000000	2016-11-29 16:31:12.0000000	(Null)	1
2	Wakil Rektor I	IT2.I	0	1	2016-11-22 11:47:16.0000000	2016-11-22 11:47:16.0000000	(Null)	1
3	Wakil Rektor II	IT2.II	0	1	2016-11-22 11:47:15.0000000	2016-11-22 11:47:15.0000000	(Null)	1
4	Wakil Rektor III	IT2.III	0	15	2016-11-22 11:47:16.0000000	2016-11-22 11:47:16.0000000	(Null)	1
5	Wakil Rektor IV	IT2.IV	0	1	2016-11-22 11:47:16.0000000	2016-11-22 11:47:15.0000000	(Null)	1
6	Ketua Senat Akademik	IT2.V	0	1	2016-11-22 11:47:15.0000000	2016-11-22 11:47:15.0000000	(Null)	2
9	Ketua Majelis Wali Amar	IT2.VI	0	(Null)	2016-11-22 11:47:15.0000000	2017-05-17 11:12:25.0000000	(Null)	3
11	Ketua Dewan Pengawas	IT2.VII	0	1	2016-11-22 11:47:16.0000000	2016-11-22 11:47:16.0000000	(Null)	4

Gambar 5.13 Data Stakeholder (Referensi)

name	code	enabled	parent_id	created_at	updated_at	level	unit_id	id
Rektor	IT2	0	(Null)	2016-11-22 11:47:15.0000000	2016-11-29 16:31:12.0000000	(Null)		1
Wakil Rektor I	IT2.I	0	1	2016-11-22 11:47:16.0000000	2016-11-22 11:47:16.0000000	(Null)		2
Wakil Rektor II	IT2.II	0	1	2016-11-22 11:47:15.0000000	2016-11-22 11:47:15.0000000	(Null)		3
Wakil Rektor III	IT2.III	0	15	2016-11-22 11:47:16.0000000	2016-11-22 11:47:16.0000000	(Null)		4
Wakil Rektor IV	IT2.IV	0	1	2016-11-22 11:47:16.0000000	2016-11-22 11:47:15.0000000	(Null)		5
Ketua Senat Akademik	IT2.V	0	1	2016-11-22 11:47:15.0000000	2016-11-22 11:47:15.0000000	(Null)		6
Ketua Majelis Wali Amar	IT2.VI	0	(Null)	2016-11-22 11:47:15.0000000	2017-05-17 11:12:25.0000000	(Null)		9
Ketua Dewan Pengawas	IT2.VII	0	1	2016-11-22 11:47:16.0000000	2016-11-22 11:47:16.0000000	(Null)		11

Gambar 5.14 Data Stakeholder (Replika) Setelah Sinkronisasi

Tabel 5.17 Jumlah Pemrosesan *Queue* pada Uji Coba 4

Delivery	Jumlah Proses	Waktu Proses (detik)
1	2	0,127249969
2	1	0,034129903
3	1	0,045481235
4	1	0,034972972
5	1	0,031919003
6	1	0,046865566
7	1	0,031229704
8	1	0,017913134
9	1	0,031101568
10	1	0,028483561
Total Per Data		0,020329704
Total Waktu Pemrosesan		0,030311210
<i>Overhead</i>		0,009981506

5.3.4.5 Skenario Uji Coba 5

Skenario uji coba 5 adalah melanjutkan sinkronisasi yang putus. Skenario uji coba 5 seperti Tabel 5.18. Hasil uji coba 5 seperti Gambar 5.11, Gambar 5.10 dan Gambar 5.12.

Tabel 5.18 Skenario Uji Coba 5

Nomor	PE-012
Nama	Melanjutkan sinkronisasi yang putus.
Tujuan	Memeriksa fungsi melanjutkan sinkronisasi yang terputus berfungsi atau tidak.
Kondisi awal	Data tabel referensi sudah terisi. Tabel replika masih kosong.
Skenario	<ol style="list-style-type: none"> 1. Administrator menjalankan perintah sinkronisasi 2. <i>Listener</i> tiba2 dimatikan. 3. <i>Listener</i> dihidupkan kembali.
Masukan	Perintah sinkronisasi.
Keluaran yang diharapkan	Bisa melanjutkan sinkronisasi walaupun koneksi terputus. Isi tabel referensi sama dengan tabel replika.
Hasil	Berhasil.

Tabel 5.19 Hasil Pemrosesan per Queue

Delivery Tag	Jumlah Proses	Waktu Proses (detik)
1	2	0,126240969
2	1	0,034689903
3	1	0,045480251
4	1	0,034977198
5	1	0,031919003
6	1	0,046765566
7	1	0,030329704
8	1	0,010913134
9	1	0,031001568
10	1	0,02878356
Total Per Data		0,421100855
Total Waktu Pemrosesan		0,462546825
<i>Overhead</i>		0,041445971

5.3.4.6 Skenario Uji Coba 6

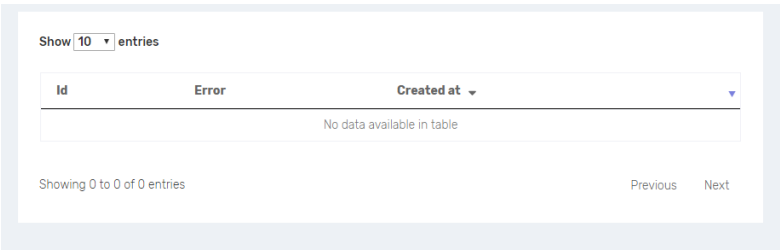
Skenario uji coba 6 merupakan skenario mendeteksi kegagalan sinkronisasi. Gagal sinkronisasi disebabkan oleh beberapa hal seperti salah *mapping* sehingga tipe data tidak sesuai atau kesalahan pada kode. Tabel 5.20 merupakan skenario pengujian uji coba 6. Gambar 5.15 menunjukkan ketidaksesuaian *mapping* dari kolom tabel referensi ke tabel replika. Kolom id (integer) ke kolom name (string), dan sebaliknya. Gambar 5.16 menunjukkan sebelum sinkronisasi dijalankan, tidak ada catatan *error*. Setelah sinkronisasi dijalankan, *listener* mengalami gagal proses sehingga Sync ITS mencatat kegagalan tersebut. Contoh hasil pencatatan kegagalan seperti Gambar 5.17.

Tabel 5.20 Skenario Uji Coba 6

Nomor	PE-013
Nama	Mendeteksi kegagalan sinkronisasi
Tujuan	Memeriksa fungsi mendeteksi kegagalan sinkronisasi berfungsi atau tidak.
Kondisi awal	1. Data tabel referensi sudah terisi. 2. Tabel replika masih kosong. 3. Tipe data kolom referensi dan kolom replika tidak sama.
Skenario	Administrator menjalankan perintah sinkronisasi
Masukan	Perintah sinkronisasi
Keluaran yang diharapkan	Bisa mendeteksi kegagalan dan mencatat kegagalan ke basis data
Hasil	Berhasil

Source Column	Target Column
id	name
name	id
code	code
enabled	enabled
parent_id	parent_id
created_at	created_at
updated_at	updated_at
level	level
unit_id	unit_id

Gambar 5.15 Mapping Tabel Referensi ke Tabel Replika



Gambar 5.16 Tampilan *Error Log* sebelum Sinkronisasi

37	[\"SQLSTATE[42S22]: [Microsoft][ODBC Driver 13 for SQL Server][SQL Server]Invalid data type 'id' and 'name.'\"]	2017-12-11 16:24:42.000
----	---	-------------------------

Gambar 5.17 Hasil Deteksi setelah Sinkronisasi

5.4 Evaluasi Pengujian

Berdasarkan pengujian fungsionalitas yang telah dilakukan, pengujian memberikan hasil yang sesuai dengan skenario yang telah direncanakan. Evaluasi pengujian pada masing-masing fungsionalitas dijelaskan sebagai berikut.

1. Pengujian membuat pengaturan koneksi baru sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-001 yang memberikan informasi bahwa proses membuat koneksi baru berjalan dengan benar.
2. Pengujian mengubah koneksi sesuai dengan yang diharapkan. Kondisi ini memperlihatkan pada pengujian PE-002 yang memberikan informasi bahwa proses mengubah pengaturan berjalan dengan benar.
3. Pengujian menghapus koneksi sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-003 yang memberikan informasi bahwa proses menghapus koneksi berjalan dengan benar.
4. Pengujian membuat pengaturan *queue* baru sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-004 yang memberikan informasi bahwa proses membuat koneksi baru berjalan dengan benar.
5. Pengujian mengubah *queue* sesuai dengan yang diharapkan. Kondisi ini memperlihatkan pada pengujian PE-005 yang memberikan informasi bahwa proses mengubah *queue* berjalan dengan benar.
6. Pengujian menghapus *queue* sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-006 yang memberikan informasi bahwa proses menghapus *queue* berjalan dengan benar.
7. Pengujian untuk memperbaiki prioritas sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-007 yang memberikan informasi bahwa proses pembaruan prioritas berjalan dengan benar.
8. Pengujian sinkronisasi dengan skenario uji coba 1 sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada

pengujian PE-008 yang memberikan informasi bahwa data yang dikirimkan sudah benar dan proses berjalan dengan benar.

9. Pengujian sinkronisasi dengan skenario uji coba 2 sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-009 yang memberikan informasi bahwa data yang dikirimkan sudah benar dan proses berjalan dengan benar.
10. Pengujian sinkronisasi dengan skenario uji coba 3 sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-010 yang memberikan informasi bahwa data yang dikirimkan sudah benar dan proses berjalan dengan benar.
11. Pengujian sinkronisasi dengan skenario uji coba 4 sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-011 yang memberikan informasi bahwa data yang dikirimkan sudah benar dan proses berjalan dengan benar.
12. Pengujian sinkronisasi dengan skenario uji coba 5 sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-012 yang memberikan informasi bahwa data yang dikirimkan sudah benar dan proses berjalan dengan benar.
13. Pengujian sinkronisasi dengan skenario uji coba 6 sesuai dengan yang diharapkan. Kondisi ini diperlihatkan pada pengujian PE-013 yang memberikan informasi bahwa Sync ITS dapat mendeteksi kegagalan dan mencatat kegagalan tersebut.

(Halaman ini sengaja dikosongkan)

BAB VI

KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dalam proses pengerjaan tugas akhir dari tahap pendahuluan, kajian pustaka, analisis, perancangan, implementasi dan pengujian kerangka kerja Sync ITS diperoleh kesimpulan sebagai berikut.

1. Sinkronisasi satu arah dari tabel replika ke tabel referensi dengan paradigma Publish/Subscribe berhasil diimplementasikan dengan membuat suatu kerangka kerja bernama Sync ITS dengan menggunakan RabbitMQ sebagai media distribusi data (*message broker*).
2. Sync ITS dapat mendeteksi perubahan menggunakan pendekatan *primary copy replication* dan *timestamp-based database synchronization* dengan membandingkan waktu *last sync* dengan *time parameter column* pada tabel referensi. Jika terdeteksi *self-reference*, maka data diurutkan menggunakan *topological sorting algorithm*. Data dibagi menjadi beberapa bagian (*chunk*) untuk di-*publish*. Hal ini dibuktikan dengan pengujian PE-008, PE-009, PE-010, PE-011, PE-012 dan PE-013.
3. Sync ITS dapat mempublikasikan perubahan data dari tabel referensi ke tabel replika dengan menggunakan paradigma *publish/subscribe*. Sync ITS mem-*publish* data ke RabbitMQ sebagai *message broker*, lalu RabbitMQ akan meneruskan data ke *subscriber*. *Subscriber* memproses data yang diterima. Apabila data ditemukan, maka perbarui data. Kalau tidak ditemukan, maka buat data baru. Hal ini dibuktikan dalam

pengujian PE-008, PE-009, PE-010, PE-011, PE-012 dan PE-013.

4. Sync ITS dapat menangani data yang gagal sinkronisasi akibat jaringan terputus dengan menggunakan fitur *publisher confirms and message acknowledgments* pada RabbitMQ. Jika *subscriber* berhasil memproses data, maka *subscriber* dapat mengirimkan pesan *ack* ke *publisher*, lalu pesan dihapus. Kalau tidak ada *ack*, maka *publisher* mengirimkan ulang pesan tersebut. Hal ini dibuktikan dengan pengujian Pe-012.

6.2 Saran

Berikut ini beberapa saran mengenai pengembangan lebih lanjut kerangka kerja Sync ITS berdasarkan hasil rancangan, implementasi, dan uji coba yang telah dilakukan.

1. Kerangka kerja Sync ITS perlu dilakukan uji coba pada basis data yang memiliki data besar dan transaksinya masif untuk menguji kebenaran.
2. Ketika *queue* dipublish ulang karena *circular*, data pada *queue* tersebut diproses ulang semua. Bisa ditingkatkan hanya proses yang *circular* saja.
3. Sync ITS perlu dikembangkan lebih lanjut agar bisa menangani perbedaan zona waktu.
4. Antarmuka kerangka kerja Sync ITS masih sederhana sehingga diperlukan pengembangan lebih lanjut.

DAFTAR PUSTAKA

- [1] C. Guyer dan C. Mansfield, “Create Linked Servers (SQL Server Database Engine),” Microsoft, 20 11 2015.[Online]. Available: <https://docs.microsoft.com/en-us/sql/relational-databases/linked-servers/create-linked-servers-sql-server-database-engine>. [Diakses 14 Juni 2017].
- [2] F. D. Munoz, H. Decker, J. E. Aremndariz dan J. R. G. d.Mendivil, “Database Replication Approaches,” Institutio Technologic de Informatica, 2007.
- [3] “Publish/Subscribe,” IBM Knowledge Center, 1 06 2017.[Online].Available:https://www.ibm.com/support/knowledgecenter/en/SSMKHH_10.0.0/com.ibm.etools.mft.doc/aq01120_.htm. [Diakses 20 Juni 2017].
- [4] “Consumer Acknowledgements and Publisher Confirms,”RabbitMQ by Pivotal, [Online]. Available: <https://www.rabbitmq.com/confirms.html>. [Diakses 20 April 2017].
- [5] T. M. Connolly dan C. Begg, A Practical Approach to Design, Implementation, and Management, fourth edition, Addison-Wesley, 2005.
- [6] Indrajani, Sistem Basis Data dalam Paket Five in One, Jakarta: PT. Elex. Media Komputindo, 2009.
- [7] Kertahadi, Sistem Informasi Manajemen, Malang: IKIP Malang, 1995.
- [8] R. G. Murdick, R. Joel E. dan J. R. Clagget, 1993, Erlanga, Sistem Informasi Untuk Manajemen Modern.
- [9] “Work Queues,” Pivotal Software. Inc, [Online]. Available: <https://www.rabbitmq.com/tutorials/tutorial-two-python.html>. [Diakses 21 Juni 2107].

- [10] “Task Scheduling,” Laravel, [Online]. Available: <https://laravel.com/docs/5.4/scheduling>. [Diakses October 2017].

BIODATA PENULIS



Muhammad Rifatullah, lahir di Lampung, pada tanggal 13 Juni 1996. Penulis menempuh pendidikan mulai dari SDN 01 Gunung Madu Lampung Tengah (2002 - 2008), SMP Satya Dharma Sudjana PT Gunung Madu Plantation Lampung Tengah (2008 - 2011), MAN 1 Bandar Lampung (2011 - 2014) dan S1 Teknik Informatika ITS (2014 - 2018).

Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTTC). Di antaranya adalah menjadi staff departemen pengembangan keprofesian (PengPro), lalu staff ahli PengPro. Penulis juga aktif dalam kegiatan kepanitiaan Schematics. Selain organisasi, penulis juga menjadi Tenaga Harian Lepas DPTSI ITS. Penulis juga merupakan vendor kontrak untuk beberapa perusahaan negara dan swasta. Komunikasi penulis dapat melalui email: **rifatullah.muhammad@gmail.com**.